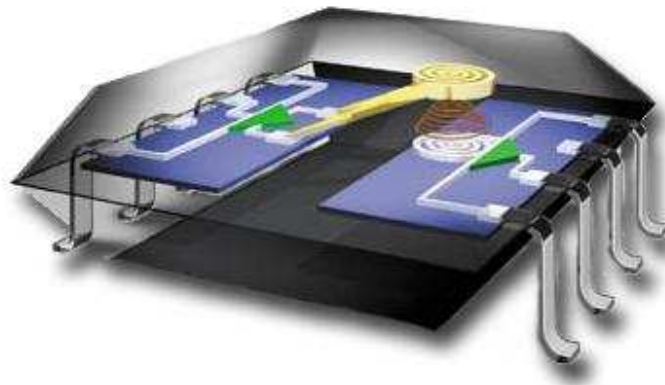


## DISEÑO DIGITAL PARA INGENIERIA



**Autor:**

**Rubén Darío Cárdenas Espinosa**  
**Matrícula Profesional CL20633345**  
[rdcardenas@gmail.com](mailto:rdcardenas@gmail.com)

**Candidato a Doctor en Ciencias con especialidad en Ingeniería Eléctrica**  
**Master of Sciences with major in Engineering Electrical de Atlantic International University**  
**Especialista en Gerencia en Finanzas, Ingeniero Electrónico, Tecnólogo Profesional en**  
**Electrónica y Automatización Industrial de Universidad Autónoma de Manizales**

**Catedrático del Programa Ingeniería de Sistemas Universidad Minuto de Dios – CERES**  
**Chinchiná**

**Gerente de Proyectos Programa Paz y Competitividad y Profesor Asistente del**  
**Departamento de Electrónica y Automatización de la Universidad Autónoma de Manizales**  
**Coordinador Ingenierías, Catedrático programas de posgrado y pregrado Universidad**  
**Antonio Nariño Sede Manizales**

**Catedrático Tecnología en Electrónica Universidad de Caldas**  
**Catedrático UNITECNICA (Ingecómputo) Manizales**

**Manizales, Departamento de Caldas**  
**República de Colombia**  
**Febrero de 2009**

### **Objetivo Terminal del Curso**

Analizar y aplicar los conceptos básicos y técnicos de diseño en sistemas lógicos combinacionales y secuenciales programables.

Implementar circuitos que controlen secuencias y procesos automáticos.

Diseñar sistemas digitales para la manipulación y procesamiento de datos.

### **Breve Descripción**

El desarrollo de la electrónica, se ha constituido en uno de los más grandes sucesos de la época moderna y ha sido fundamental para los grandes adelantos tecnológicos, en casi todos los campos del saber.

Las herramientas y conocimientos de electrónica digital son necesarios para el análisis, diseño e implementación de circuitos y sistemas digitales, en especial con el almacenamiento, transformación y comunicación de la información en forma digital.

## TABLA DE CONTENIDO

<b>Presentación</b>	5
<b>1. Sistemas Numéricos</b>	6
1.1. Sistema Decimal	7
1.2. Sistema Binario	7
1.3. Sistema Hexadecimal	8
1.4. Sistema Octal	8
1.5. Conversiones de un Sistema a Otro	10
1.6. Representación de Números Enteros y de Punto Flotante	13
1.7. Operaciones Aritméticas en Binario	16
<b>2. Principios de Diseño de Lógica Combinacional</b>	20
2.1. ¿Qué es Electrónica Digital?	20
2.2. Álgebra de Boole	20
a. Operaciones Booleanas y Compuertas Básicas	21
b. Compuertas Lógicas Combinadas	22
c. Circuitos Integrados y Circuito de Prueba	24
d. Propiedades de las Operaciones Booleanas	25
e. Teoremas Boléanos	27
f. Teoremas de DeMorgan	27
2.3. Simplificación de Expresiones Lógicas	30
2.4. Implementación de Funciones Lógicas mediante Compuertas	31
2.5. Síntesis de Diseño de Circuitos Combinacionales	31
2.6. Métodos para Sintetizar Circuitos Lógicos	32
a. Método de Suma de Productos (SDP)	34
b. Método de producto de sumas (PDS)	37
c. Mapas de Karnaugh	39
d. Algoritmo de Quine – McCluskey	47
<b>3. Circuitos Lógicos Combinacionales</b>	48
3.1. Circuitos Aritméticos	50
3.2. Decodificadores	67
3.3. Registros de Tres Estados	75
3.4. Codificadores	76
3.5. Multiplexores y Demultiplexores	81
3.6. Generadores de Paridad	87
3.7. Comparadores	91
3.8. Implementación de Funciones Lógicas con Decodificadores	95
3.9. Implementación de Funciones Lógicas con Multiplexores	96
<b>4. Lógica Secuencial</b>	99
4.1. Oscilador Simétrico con compuertas NOT	100
4.2. Disparadores Schmitt Trigger	101
4.3. Oscilador de Cristal	102
4.4. Osciladores Controlados	104
4.5. Circuito Integrado 555	104
4.6. CI 555 como Multivibrador Astable	106
4.7. CI 555 como Multivibrador Monoestable	106
4.8. Circuitos Monoestables	107
4.9. Circuitos Biestables (FLIP-FLOPs)	108
4.9.1. FLIP-FLOP Básico R-S (Reset-Set)	108
4.9.2. FLIP FLOP RS - Controlado por un pulso de reloj	109
4.9.3. FLIP-FLOP D	110
4.9.4. FLIP-FLOP D PRESET-CLEAR	112
4.9.5. FLIP-FLOP J-K	113

4.9.6. FLIP-FLOP T (Toggle)	113
<b>5. Contadores y Registros</b>	114
5.1. Contadores de Propagación	114
5.2. Contador de propagación ascendente	114
5.3. Contadores con números $\text{MOD} < 2^n$	116
5.4. Contador de propagación descendente	116
5.5. Contadores Sincrónicos	117
5.6. Ejemplos de Contadores en Circuito Integrado	120
5.7. Registros de Corrimiento	123
5.8. Registro de Corrimiento Básico	123
5.9. Tipos de Entradas y Salidas en los Registros de Corrimiento	124
5.10. Registros de corrimiento bidireccionales	125
5.11. Registros en Circuito Integrado	126
5.12. Aplicaciones de los Registros de Corrimiento	128
5.13. Contador en Anillo	128
<b>6. Análisis y Diseño de Circuitos Secuenciales</b>	130
6.1. Teoría de Máquinas de Estado (FSM)	130
6.2. Máquinas de Estado de Mealy y Moore	130
6.3. Ecuaciones Lógicas	132
6.4. Tablas de Estado	132
6.5. Diagramas de Estado	134
6.6. Tablas de Transición de FLIP-FLOPs	134
6.7. Mapas de Karnaugh	135
6.8. Análisis y Diseño de Circuitos Secuenciales Sincrónicos	136
6.9. Diseño de Circuitos Secuenciales con FLIP-FLOPs D	139
6.10. Estados no usados	141
6.11. Análisis de Circuitos Secuenciales Asincrónicos	144
6.12. Ejemplos de Control Secuencial	149
<b>7. Dispositivos Lógicos Programables</b>	154
7.1. Tipos de PLD	155
7.2. Estructura de los Dispositivos Lógicos Programables Básicos	155
7.3. Herramientas para la Automatización del Diseño Electrónico	156
7.4. Principios y Aplicaciones de los Dispositivos Lógicos Programables como las PALs y las GALs.	157
7.5. Arquitectura de Diversos PLD's Secuenciales	163
7.6. Memorias	
7.7. Aplicaciones de las Memorias	177
7.8. Lógica programable temprana	185
Bibliografía	186
ANEXO 1 Evaluación Diseño Electrónico Digital	187
ANEXO 2 Respuestas Evaluación Diseño Electrónico Digital	192
ANEXO 3 Proyecto Conversión análoga digital	193

## PRESENTACIÓN

Los circuitos digitales se emplean en productos electrónicos como videojuegos, hornos de microondas, sistemas de control para automóviles, dispositivos biomédicos, entre otros; también los podemos encontrar equipos de prueba como medidores, generadores y osciloscopios, dispositivos de telecomunicación y consumo masivo como los celulares, radios, televisores y computadores personales.

La era de la electrónica Digital está en auge y las técnicas digitales han reemplazado a muchos de los circuitos análogos empleados en el pasado.

El papel o finalidad de la Educación es generar valores y conocimientos que permitan convertir a todos los individuos en seres capaces de pensar, sentir, realizar, generar e innovar, para contribuir al mejoramiento y beneficio de la sociedad a la cuál pertenecen.

La Educación abierta o a distancia apoyada en tecnología no se debe limitar al conocimiento, debe asumir el reto de desarrollar las herramientas necesarias que le permitan al estudiante ser un participante activo de su proceso de aprendizaje, éste debe ser inquieto, preguntón crítico y motivarse cada vez más en lo que aprende de su profesor y motivar en él un sentimiento investigativo e innovador.

El enfoque de este tipo de Educación está orientado al estudiante, quién es el directo responsable de su proceso de aprendizaje, por lo tanto, para el desarrollo de este curso se implementa un modelo pedagógico y andragógico de aprendizaje interactivo virtual con el apoyo técnico necesario que apoye la Psicología de Aprendizaje, haciendo énfasis en sus tres elementos: Perdurabilidad, Transparencia y Habilidad Práctica. Esto permitirá contrarrestar las desventajas que se presentan en la Educación virtual como son la desmotivación y deserción del estudiante por la lentitud del proceso y por la reducción de la interacción personal.

Para lograr aprendizajes sistémicos en la educación, es necesario avanzar hacia un modelo de pedagogía y andragogía que se identifique con los aspectos sociales que se espera, impacten, para lograr así, una transformación de la realidad a través de la educación.

El presente curso ha sido elaborado a partir de mi experiencia docente y profesional aplicada en la Universidad Autónoma de Manizales, Universidad de Caldas, Universidad Antonio Nariño, Unitécnica (Ingecómputo) y otras Instituciones del Eje Cafetero.

El Módulo Diseño Digital para Ingeniería brindará a los estudiantes los conceptos y técnicas empleadas en el diseño de sistemas lógicos combinacionales y secuenciales, las herramientas y conocimientos de los Circuitos Digitales, a través de ejercicios de aplicación para afianzar los conceptos vistos, y el aprendizaje significativo, facilitando transferir los conocimientos adquiridos a otros contextos de su quehacer profesional.

**Dedico este trabajo a mi bella esposa Kathy Faridy, a mi bebé por nacer Sara Gabriela, mi mamá Ofelia, tías Lilia y Lucila y demás miembros de mi familia y en especial a la memoria de mi tío José Hernán Espinosa Martínez** quien fue un padre para mí (mi mentor) y le debo lo que soy hoy día **“que la luz de su alma nos siga guiando por el camino de la vida”**.

**Rubén Darío**

## 1. Sistemas Numéricos

El sistema decimal es universalmente empleado para representar cantidades en el mundo real. Los sistemas electrónicos digitales tienen que recoger la información y convertirla en dígitos binarios para procesarla internamente. Así mismo, cuando la información es procesada, es necesario convertir esta información, por lo general a decimal antes de llevarla al mundo exterior. En realidad, no se manejan solamente estos dos sistemas, en la práctica se hace necesario utilizar códigos que facilitan el manejo de otras características. En este capítulo, se describirá el código decimal, el código binario, el hexadecimal, el octal, las operaciones entre estos sistemas, las distintas conversiones entre los diferentes sistemas y algunas representaciones de números binarios.

### Sistemas Binario y Hexadecimal

El sistema binario es el más utilizado en los circuitos electrónicos digitales. Existen otros dos sistemas, en las aplicaciones digitales; El hexadecimal y el octal. Su ventaja radica en la facilidad que ofrecen para representar de forma reducida los números binarios.

#### 1.1. Sistema Decimal

El sistema decimal es un sistema en base 10. En una cantidad decimal cada dígito tiene un peso asociado a una potencia de 10 según la posición que ocupe. Los pesos para los números enteros son potencias positivas de diez, aumentado de derecha a izquierda, comenzando por  $10^0=1$ .

$$\text{Peso:} \dots 10^6 10^5 10^4 10^3 10^2 10^1 10^0$$

Los pesos para los números fraccionarios son potencias negativas de diez, aumentando de izquierda a derecha, comenzando por  $10^{-1}$ .

$$\text{Peso:} \dots 10^6 10^5 10^4 10^3 10^2 10^1 10^0, 10^{-1} 10^{-2} 10^{-3} 10^{-4}$$

La expresión general para descomponer el valor de una magnitud expresada en cualquier sistema numérico para obtener su valor decimal:

$$\sum_{i=-n}^{p-1} d_i \times r^i \quad \text{donde,}$$

$d_i$  = Dígito en la posición  $i$ .

$r$  = Base del sistema utilizado.

$n$  = No. de dígitos fraccionarios.

$p$  = No. de dígitos enteros.

La base  $r$  del sistema numérico es el número total de dígitos permitidos para el sistema.

$$\text{Ejemplo } 235.63 = 2 \times 10^2 + 3 \times 10^1 + 5 \times 10^0 + 6 \times 10^{-1} + 3 \times 10^{-2}$$

## 1.2. Sistema Binario

El sistema binario es un sistema en base dos. Es el sistema utilizado por los computadores digitales y tiene sólo dos valores lógicos posibles - "0 y 1" - para sus coeficientes, los cuales se pueden representar físicamente de distintas maneras, como las siguientes:

- Tensiones alto y bajo.
- Interruptor cerrado o abierto.
- Sentido de magnetización de un núcleo magnético.
- Corriente eléctrica alta o baja.

Los dígitos 0 y 1 se llaman bits.

En un número entero binario el BIT a la derecha es el BIT menos significativo (*LSB, Least Significant Bit*) y tiene un peso de  $2^0=1$ . El BIT del extremo izquierdo el BIT más significativo (*MSB, Most Significant Bit*) y tiene un peso dependiente del tamaño del número binario. Los pesos crecen de derecha a izquierda en potencias de 2. En números fraccionarios el bit a la izquierda de la coma es el *MSB* y su peso es de  $2^{-1}=0,5$ . Los pesos decrecen de izquierda a derecha en potencias negativas de 2.

$$\text{Peso: } 2^{n-1} \dots 2^4 2^3 2^2 2^1 2^0, 2^{-1} 2^{-2} 2^{-3} \dots 2^{-n}.$$

En el cual n es el número de bits a partir de la coma binaria. La tabla 1.1.1. muestra la equivalencia de los números decimales del 0 al 15 a su correspondiente binario.

Número Decimal	Número Binario			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

**Tabla 1.** Sistema decimal y binario

Ejemplo  $101101,11 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$

En decimal se tiene:  $32 + 8 + 4 + 1 + 0,5 + 0,25 = 45,75_{10}$ .

### 1.3. Sistema Hexadecimal

El sistema hexadecimal es un sistema en base 16 y consta de 16 dígitos diferentes que son: del 0 al 9 y luego de la letra A a la F, es decir 10 dígitos numéricos y seis caracteres alfabéticos.

El sistema hexadecimal se usa como forma simplificada de representación de números binarios y debido a que 16 es una potencia de 2 ( $2^4=16$ ), resulta muy sencilla la conversión de los números del sistema binario al hexadecimal y viceversa.

La tabla 2. Muestra los números decimales de 0 al 15 con su equivalencia en binario y hexadecimal.

Decimal	Sistema binario	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

**Tabla 2.** Sistema decimal, binario y hexadecimal

Para convertir un número hexadecimal en un número binario se reemplaza cada símbolo hexadecimal por un grupo de cuatro bits.

*Ejemplo:*

El número  $4F5B_{16}$  en binario equivale a

0100111101011011  
 └─┘ └─┘ └─┘ └─┘  
 4    F    5    B

### 1.4. Sistema Octal

El sistema octal es un sistema en base 8 y está formado por 8 dígitos. En un número octal, los pesos crecen de derecha a izquierda en potencias de 8.

$$\text{Peso: } 8^4 8^3 8^2 8^1 8^0$$

La tabla 3. Muestra los números decimales de 0 al 17 con su equivalencia a binario y octal.



Decimal	Sistema binario	Octal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	10
9	1001	11
10	1010	12
11	1011	13
12	1100	14
13	1101	15
14	1110	16
15	1111	17
16	10000	20
17	10001	21

**Tabla 3.** Sistema decimal, binario y octal

Observe que en octal los dígitos 8 y 9 no se usan. La conversión de un número octal en decimal se obtiene multiplicando cada dígito por su peso y sumando los productos.

*Ejemplo:*  $1725 = 1 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 5 \times 8^0 = 512 + 448 + 16 + 5 = 981$

### Código decimal binario (BCD)

El código decimal binario (*BCD Binary Code Decimal*) es utilizado para expresar los diferentes dígitos decimales con un código binario. Por consiguiente, el código *BCD* tiene diez grupos de código y resulta práctico para convertir entre decimal y *BCD*.

**El código 8421:** Pertenece al grupo de códigos *BCD*. El nombre *8421* indica los diferentes pesos de los cuatro bits binarios ( $2^3$ ,  $2^2$ ,  $2^1$ ,  $2^0$ ). La tabla 4. Muestra los números decimales de 0 al 9 con su equivalencia en *BCD*.

Decimal	Dígito en BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

**Tabla 4.** Sistema decimal y BCD

Con un número de 4 bits se pueden representar  $2^4$  combinaciones posibles, pero al emplear el código *8421* se incluyen solamente 10 grupos de código binario, en consecuencia las combinaciones *1010*, *1011*, *1100*, *1101*, *1110*, *1111* no se utilizan.

*Ejemplo:* Convertir a BCD el número decimal 6498.

Reemplazando por los valores de la tabla 4. Se obtiene,

$$6498_{10} = (0110\ 0100\ 1001\ 1000)_{8421}$$

### 1.5. Conversiones de un Sistema a Otro

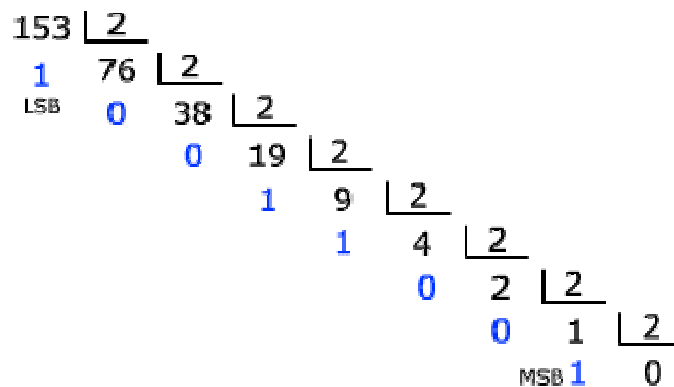
Las conversiones entre números de bases diferentes se efectúan por medio de operaciones aritméticas simples. Dentro de las conversiones más utilizadas se encuentran:

#### 1.5.1. Conversión de Decimal a Binario

Para la conversión de decimal a binario se emplean dos métodos. El primero es divisiones sucesivas y el segundo es suma de potencias de 2.

**Por divisiones sucesivas:** Se va dividiendo la cantidad decimal por 2, apuntando los residuos, hasta obtener un cociente cero. El último residuo obtenido es el BIT más significativo (MSB) y el primero es el BIT menos significativo (LSB).

Ejemplo Convertir el número  $153_{10}$  a binario.



**Figura 1.** Ejemplo de conversión de decimal a binario

El resultado en binario de  $153_{10}$  es 10011001

**Por sumas de potencias de 2:** Este método consiste en determinar el conjunto de pesos binarios cuya suma equivalga al número decimal.

Ejemplo: Convertir el número  $153_{10}$  a binario.

$$153_{10} = 2^7 + 2^4 + 2^3 + 2^0 = 128 + 16 + 8 + 1; \quad 153_{10} = 10011001_2$$

#### 1.5.2. Conversión de Fracciones Decimales a Binario

Para la conversión de fracciones decimales a binario se emplean el siguiente método.

**Por suma de potencias de 2:** Emplea la misma metodología de la suma de potencias de 2 pero se trabaja con potencias negativas.

Ejemplo Convertir el número  $0,875_{10}$  a binario.

$$0,875_{10} = (2^{-1}) + (2^{-2}) + (2^{-3}) = 0,5 + 0,25 + 0,125 = 0,111_2$$

**Por multiplicaciones sucesivas:** La conversión de números decimales fraccionarios a binario se realiza con multiplicaciones sucesivas por 2. El número decimal se multiplica por 2, de éste se extrae su parte entera, el cual va a ser el *MSB* y su parte fraccional se emplea para la siguiente multiplicación y seguimos sucesivamente hasta que la parte fraccional se vuelva cero o maneje un error moderado. El último residuo o parte entera va a constituir el *LSB*.

Ejemplo: Convertir el número  $0,875_{10}$  a binario.

Número N	N X 2	Parte entera	Peso
0,875	1,75	1	MSB
0,75	1,5	1	
0,5	1,00	1	LSB

**Tabla 5.** Ejemplo de Conversión de Decimal a Binario.

El resultado en binario de  $0,875_{10}$  es  $0,111_2$ .

### 1.5.3. Conversión de Decimal a Hexadecimal

En la conversión de una magnitud decimal a hexadecimal se realizan divisiones sucesivas por 16 hasta obtener un cociente de cero. Los residuos forman el número hexadecimal equivalente, siendo el último residuo el dígito más significativo y el primero el menos significativo.

Ejemplo: Convertir el número  $1869_{10}$  a hexadecimal.

$$\begin{array}{r}
 1869 \overline{) 16} \\
 \underline{13} \phantom{00} \\
 116 \overline{) 16} \\
 \underline{4} \phantom{00} \\
 7 \overline{) 16} \\
 \underline{0} \\
 \text{MSB}
 \end{array}$$

D

**Figura.2.** Ejemplo de Conversión de decimal a hexadecimal

El resultado en hexadecimal de  $1869_{10}$  es  $74D_{16}$ .

### 1.5.4. Conversión de Decimal a Octal

En la conversión de una magnitud decimal a octal se realizan divisiones sucesivas por 8 hasta obtener la parte entera del cociente igual a cero. Los residuos forman el número octal equivalente, siendo el último residuo el dígito más significativo y el primero el menos significativo.

Ejemplo: Convertir el número  $465_{10}$  a octal.

Número N	N ÷ 8	Parte decimal	Parte decimal x 8	Peso
465	58,125	0,125	1	LSB
58	7,25	0,25	2	
0,5	0,875	0,875	7	MSB

**Tabla 6.** Ejemplo de Conversión de Decimal a Hexadecimal.

El resultado en octal de 465<sub>10</sub> es 721.

### 1.5.5. Conversión de Binario a Decimal

Un número binario se convierte a decimal formando la suma de las potencias de base 2 de los coeficientes cuyo valor sea 1.

Ejemplo: Convertir el número 1100<sub>2</sub> a decimal.

$$1100_2 = 1 \times 2^3 + 1 \times 2^2 = 12_{10}$$

### 1.5.6. Conversión de Binario a Hexadecimal

El método consiste en conformar grupos de 4 bits hacia la izquierda y hacia la derecha del punto que indica las fracciones, hasta cubrir la totalidad del número binario. Enseguida se convierte cada grupo de número binario de 4 bits a su equivalente hexadecimal.

Ejemplo: Convertir el número 10011101010 a hexadecimal.

$$\begin{array}{|c|c|c|c|} \hline 0 & 1001 & 1101 & 010 \\ \hline 4 & E & A & \\ \hline \end{array} = 4EA_{16}$$

### 1.5.7. Conversión de Binario a Octal

El método consiste en hacer grupos de 3 bits hacia la izquierda y hacia la derecha del punto que indica las fracciones, hasta cubrir la totalidad del número binario. Enseguida se convierte cada grupo de número binario de 3 bits a su equivalente octal.

$$\begin{array}{|c|c|c|} \hline 001 & 010 & 101 \\ \hline 1 & 2 & 5 \\ \hline \end{array} = 125_8$$

Ejemplo: Convertir el número 01010101<sub>2</sub> a octal.

### 1.5.8. Conversión de Hexadecimal a Decimal

En el sistema hexadecimal, cada dígito tiene asociado un peso equivalente a una potencia de 16, entonces se multiplica el valor decimal del dígito correspondiente por el respectivo peso y realizar la suma de los productos.

Ejemplo Convertir el número 31F<sub>16</sub> a decimal.

$$31F_{16} = 3 \times 16^2 + 1 \times 16 + 15 \times 16^0 = 3 \times 256 + 16 + 15 = 768 + 31 = 799_{10}$$

### 1.5.9. Conversión de Hexadecimal a Binario

La conversión de hexadecimal a binario se facilita porque cada dígito hexadecimal se convierte directamente en 4 dígitos binarios equivalentes.

*Ejemplo:* Convertir el número  $1F0C_{16}$  a binario.

$$1F0C_{16} = 1111100001100_2$$

### 1.5.10. Conversión de Octal a Decimal

La conversión de un número octal a decimal se obtiene multiplicando cada dígito por su peso y sumando los productos:

*Ejemplo:* Convertir  $4780_8$  a decimal.

$$4780 = (4 \times 8^3) + (3 \times 8^2) + (8 \times 8^1) + (0 \times 8^0) = 2048 + 192 + 64 + 0 = 2304$$

### 1.5.11. Conversión de Octal a Binario

La conversión de octal a binario se facilita porque cada dígito octal se convierte directamente en 3 dígitos binarios equivalentes.

*Ejemplo:* Convertir el número  $715_8$  a binario.  $715_8 = (111001101)_2$

**1.6. Representación de Números Enteros y de Punto Flotante:** Los computadores deben interpretar números positivos y negativos. Los números binarios se caracterizan por su magnitud y su signo. El signo indica si el número es positivo o negativo y la magnitud el valor del número.

**1.6.1. Representación de Números Binarios Enteros:** Existen tres formas de representar los números binarios enteros con signo:

- Signo – magnitud.
- Complemento a 1.
- Complemento a 2.

#### **a. Signo – Magnitud**

En el sistema Signo – Magnitud los números positivos y negativos tienen la misma notación para los bits de magnitud pero se diferencian en el bit del signo. El bit del signo es el bit situado más a la izquierda en el número binario:

- En números positivos se emplea el bit "0".
- En números negativos se emplea el bit "1".
- El número no debe estar complementado.

*Ejemplo:* El número decimal 21 se expresa en binario de 6 bits 010101, donde el primer bit "0" denota el bit de una magnitud positiva. El número decimal -21 se expresa en binario 110101, donde el primer bit "1" denota el bit de una magnitud negativa.

#### **b. Complemento a 1:**

El complemento a 1 en binario se obtiene cambiando los unos por ceros y los ceros por unos. La representación de números positivos en complemento a 1 sigue las mismas reglas del sistema signo-magnitud y la representación de los números negativos en complemento 1 es el complemento a 1 del número positivo.

*Ejemplo:* El número decimal 21 se expresa en complemento a 1 a 6 bits como 010101, donde el primer bit "0" denota el bit de una magnitud positiva.

El complemento 1 a 6 bits del decimal  $-21$ , se obtiene por medio del complemento a 1 del número positivo 010101 el cual es 101010.

### c. Complemento a 2

Los computadores utilizan la representación binaria en complemento a 2 para representar números negativos. La representación de números positivos en complemento a 2 sigue las mismas reglas del sistema signo-magnitud y la representación de los números negativos en complemento a 2 se obtiene de la siguiente forma:

1. Se representa el número decimal dado en magnitud positiva.
2. El número de magnitud positiva se representa en forma binaria positiva.
3. Se obtiene el complemento 1 del número binario obtenido en el paso anterior mediante el cambio de los unos por ceros y viceversa.
4. Al complemento 1 se le suma uno y el resultado es la representación en el complemento 2.

*Ejemplo:* Representar el número  $-5_{10}$  en binario, utilizando el complemento a 2 con 5 bits.

1.  $-5 \rightarrow 5$ .
2. Escribimos el número  $+5_{10}$  en binario de 5 bits **0101**
3. Obtenemos el complemento a 1 de 0101 **1010**
4. Al complemento de número anterior se la suma 1. El resultado es **1011**.
5. Obtenemos el número 1011 en complemento a 2.

*Ejemplo:* Obtener el complemento a 2 del número positivo de 8 bits  $00000101_2 (+5_{10})$ .

El equivalente en complemento a 1 es 11111010. El complemento a 2 del número es 11111011. Comprobando los pesos en decimal se puede demostrar la obtención del negativo del número inicial utilizando el método del complemento a 2:

$$11111011_2 = (-128 + 64 + 32 + 16 + 8 + 0 + 2 + 1)_{10} = -5_{10}$$

En la representación en complemento 2 el primer bit del lado más significativo puede interpretarse como el signo, siendo cero para números positivos y 1 para números negativos. Se puede comprobar que si a una cantidad negativa expresada en complemento 2 se le saca su complemento 2, se obtiene la magnitud positiva correspondiente.

### 1.6.2. Representación en Punto Fijo y en Punto Flotante

En los computadores los números se representan en punto fijo y en punto flotante.

**Punto fijo:** Se usa para los números enteros con signo o fracciones con signo. En este caso las cantidades se representan en forma binaria en complemento a 1 ó a 2 y se pueden utilizar longitudes de 8, 16 y 32 bits. En 8 bits el rango va desde 128 hasta 127. El número de combinaciones diferentes de un número binario de  $n$  bits es:

$$N^{\circ} \text{ total de combinaciones: } 2^n.$$

En los números con signo e complemento a 2, el rango de valores para números de  $n$  bits:

$$(2^{n-1}) a + (2^{n-1}-1).$$

#### a. Enteros con signo

Los enteros de punto fijo usan un punto binario a la derecha del *LSB*.

Ejemplo El número de punto fijo de 8 bits 01110101 en complemento a 2, por tener un 0 en el bit de signo representa:

El número entero positivo 1110101 ó la fracción positiva 0.1110101

#### b. Fracciones de punto fijo

Las fracciones de punto fijo usan el punto binario entre el bit de signo y el *MSB*.

Ejemplo: El número de punto fijo de 8 bits 11001111 en complemento a 2, por tener un 1 en el bit de signo representa: El número entero negativo -0110001 ó la fracción negativa -0.0110001.

**Punto flotante:** El punto flotante se utiliza para representar números no enteros, números muy grandes o números muy pequeños.

Un número en punto flotante se expresa como  $m \times r^e$  donde,  $m$  es la mantisa y es un número de punto fijo,  $e$  es el exponente o característica y es un entero de punto fijo,  $r$  es la base. En los computadores personales se usa base 2.

La mantisa representa la magnitud del número. El exponente es la parte que representa el número de lugares a desplazar el punto decimal o binario.

Sí tenemos un número de punto fijo de la forma  $\pm (a_{n-1} \dots a_0 . a_{-1} \dots a_{-m})_r$  en forma de punto flotante será de la forma  $\pm (a_{n-1} \dots a_{-m})_r \times r^n$ , la base generalmente se omite.

Con frecuencia la mantisa  $m$  se escribe con magnitud y signo de la siguiente forma, y en forma de fracción  $M = (s_m . a_{n-1} \dots a_{-m})$  donde,  $s_m$  indica el signo (1 para una cantidad negativa y 0 para una cantidad positiva) y  $a_{n-1} \dots a_{-m}$  **representa la magnitud**.

Un número de punto flotante está normalizado si el exponente se ajusta de modo que la mantisa tenga un valor distinto de cero en la posición más significativa.

Ejemplo: El número +1010.0111 en representación normalizada en punto flotante da como resultado

$$(0.10100111) \times 2^4$$

El estándar ANSI/IEEE 754-1985 define tres formatos para los números de punto flotante:

- Precisión sencilla: Utiliza 32 bits.
- Doble precisión: Utiliza 64 bits
- Precisión ampliada: Utiliza 80 bits.

Ejemplo: Un formato a 32 bits es el siguiente, El exponente desplazado se obtiene adicionando 127 al exponente real y convirtiéndolo al binario correspondiente.

### 1.7. Operaciones Aritméticas en Binario

Los circuitos de control básicos y los computadores efectúan operaciones aritméticas. Estas operaciones se realizan en sistema binario y las leyes que las rigen, son paralelas a las usadas en el sistema decimal. A continuación se describe cada una de las metodologías para realizar tales operaciones.

#### a. Suma Binaria

La suma de dos cantidades binarias empieza con la suma de los dos dígitos menos significativos de los sumandos y un acarreo inicial de cero ó uno (Acarreo  $C_{in}$ ). Esta operación puede producir un bit de acarreo (Acarreo  $C_{out}$ ) para la suma de la siguiente posición significativa. En la tabla 7. Las entradas  $A$ ,  $B$  y  $C_{in}$  denotan al primer sumando, el segundo sumando y el acarreo de entrada. Las salidas  $S$  y  $C_{out}$  representan a la suma y el acarreo de salida.

Sumando A	Sumando B	Acarreo $C_{in}$	Acarreo $C_{out}$	Suma S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

**Tabla 7.** Suma binaria



*Ejemplo:* Efectuar la suma de 010110 y 101010.

1 1 1 1 1	Acarreo	Comprobación en decimal:
0 1 0 1 1 0		22
+ 1 0 1 0 1 0		+ 42
1 0 0 0 0 0 0		64 (2 <sup>6</sup> )

La suma de 2 magnitudes binarias en representación de complemento a 2, da como resultado la suma binaria en complemento a 2.

### b. Resta Binaria:

En la resta binaria, los bits del minuendo de las columnas se modifican cuando ocurre un préstamo. En la tabla 1.4.2. las entradas  $A$ ,  $B$  y  $B_{in}$  denotan el minuendo, el sustraendo y el bit prestado. Las salidas  $D$  y  $P$  representan a la diferencia y el préstamo. La tabla muestra los resultados de una resta binaria de dos bits,

Minuendo A	Sustraendo B	Préstamo $B_{in}$	Préstamo P	Diferencia D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

**Tabla 8.** Resta binaria

Para  $A=0$ ,  $B=0$  y  $B_{in}=1$ , hay que tomar prestado un 1 de la siguiente columna más significativa, lo cual hace  $P=1$  y agregar "en decimal" 2 a A. La resta  $2-0-1=1$ , da como resultado en binario  $D=1$ .

Los préstamos se propagan hacia la izquierda de columna en columna.

*Ejemplo:* Restar  $1001_2$  de  $10011_2$ .

Renglón 2, Tabla 1.4.1.  $0 - 1 = 0$  con un préstamo de la columna izquierda.  $10 - 1 = 1$

Renglón 1, Tabla 1.4.1.  $0 - 0 = 0$  sin préstamo.

Renglón 3, Tabla 1.4.1.  $1 - 0 = 0$  sin préstamo.

↓ Renglón 4, Tabla 1.4.1.  $1 - 1 = 0$  sin préstamo.

1	↓	Préstamo
1 0 0 1 1		
- 0 1 0 0 1		
0 1 0 1 0		

**Rebasamiento:** Se presenta cuando la suma de la columna más significativa genera un acarreo. Este sólo se puede producir cuando ambos números son positivos o negativos.

*Ejemplo:* Efectuar la suma de  $865_{10}$  y  $412_{10}$ .

$$\begin{array}{r}
 1 \quad \text{Acarreo} \\
 865 \\
 + 412 \\
 \hline
 1207 \\
 \uparrow \\
 \text{Rebasamiento}
 \end{array}$$

*Ejemplo:* Efectuar la suma de  $110_2$  y  $110_2$ .

$$\begin{array}{r}
 11 \quad \text{Acarreo} \\
 110 \\
 + 110 \\
 \hline
 1100 \\
 \uparrow \\
 \text{Rebasamiento}
 \end{array}$$

## Resta binaria en Complemento a 2

En la lección anterior se vió que el signo de un número positivo ó negativo se cambia calculando su complemento a 2. La resta de dos números con signo se calcula sumando el complemento a 2 del sustraendo al minuendo y descartando cualquier bit de acarreo final. El siguiente procedimiento es necesario para calcular la resta de dos números:

1. Obtener el complemento a 2 del sustraendo.
2. Efectuar la suma del minuendo y el sustraendo en complemento a 2.
3. Sí la suma presenta rebasamiento indica que la respuesta es positiva. Ignore el rebasamiento.
4. Si no hay rebasamiento, entonces la respuesta es negativa. Para obtener a magnitud del número binario, obtenga el complemento a dos de la suma.

*Ejemplo:* Sustraer  $(1010111 - 1001000)_2$

1. El complemento a 2 de  $1001000$  es  $0111000$ .
2. Sumamos el primer sumando y el complemento a 2 obtenido.

111	Acarreo	Comprobación en decimal:
1010111		87
+ 0111000		- 72
10001111		15
↑		
Rebasamiento (Se ignora )		

3. La respuesta es  $0001111_2$ .

### Multiplicación Binaria

La multiplicación de dos cantidades binarias es necesario considerar lo siguiente:

Multiplicando A	Multiplicador B	Multiplicación (A*B)
0	0	0
0	1	0
1	0	0
1	1	1

**Tabla 8.** Multiplicación binaria

La multiplicación binaria cumple las mismas reglas de la multiplicación decimal. En el próximo ejemplo se ilustrará la multiplicación binaria.

*Ejemplo:* Multiplicar las cantidades 1011 y 1101.

$$\begin{array}{r}
 \begin{array}{cccc}
 A_3 & A_2 & A_1 & A_0 \\
 B_3 & B_2 & B_1 & B_0
 \end{array} \\
 \hline
 & & & B_0A_3 & B_0A_2 & B_0A_1 & B_0A_0 \\
 & & & B_1A_3 & B_1A_2 & B_1A_1 & B_1A_0 \\
 & & B_2A_3 & B_2A_2 & B_2A_1 & B_2A_0 \\
 & B_3A_3 & B_3A_2 & B_3A_1 & B_3A_0 \\
 \hline
 P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0
 \end{array}$$

**Figura 3.** Multiplicación binaria

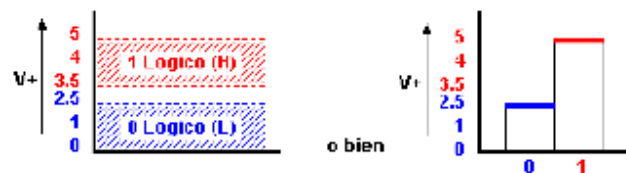
**Multiplicación con signo:** Se representan los operandos en complemento 2 y el resultado también se obtiene en complemento 2. El último multiplicando desplazado se niega.

## 2. Principios de Diseño de Lógica Combinacional

### 2.1. ¿Qué es Electrónica Digital?

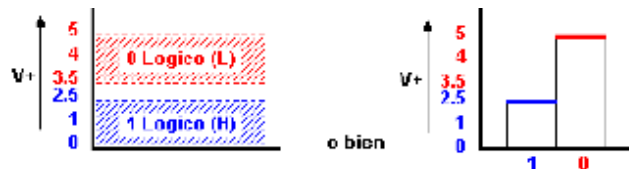
Es una rama de la ciencia aplicada que estudia las señales eléctricas, pero en este caso son señales discretas, es decir, están bien identificadas, razón por la cual a un determinado nivel de tensión se lo llama estado alto (High) o Uno lógico; y a otro, estado bajo (Low) o Cero lógico, por lo tanto, existe la Lógica Positiva y la Lógica Negativa,

**Lógica Positiva:** En esta notación al 1 lógico le corresponde el nivel más alto de tensión (positivo) y al 0 lógico el nivel más bajo (que bien podría ser negativo), pero, *¿que ocurre cuando la señal no está bien definida?* Entonces habrá que conocer cuales son los límites para cada tipo de señal (conocido como tensión de histéresis), en la figura 4 se puede ver con mayor claridad cada estado lógico y su nivel de tensión.



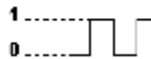
**Figura 4.** Estado Lógico Positivo y Nivel de Tensión

**Lógica Negativa:** Aquí ocurre todo lo contrario, es decir, se representa al estado "1" con los niveles más bajos de tensión y al "0" con los niveles más altos. (Ver Figura 5).



**Figura 5.** Estado Lógico Negativo y Nivel de Tensión

Por lo general se suele trabajar con lógica positiva, y así lo haremos en este curso, la forma más sencilla de representar estos estados es como se puede ver en la figura 6.



**Figura 6.** Estado Lógico Positivo

### 2.2. Álgebra de Boole

El álgebra Booleana es la teoría matemática que se aplica en la lógica combinatoria. Las variables Booleanas son símbolos utilizados para representar magnitudes lógicas y pueden tener sólo dos valores posibles: 1 (valor alto) ó 0 (valor bajo).

### a. Operaciones Booleanas y Compuertas Básicas

Las operaciones booleanas son posibles a través de los operadores binarios negación, suma y multiplicación, es decir que estos combinan dos o más variables para conformar funciones lógicas.

Una compuerta es un circuito útil para realizar las operaciones anteriormente mencionadas, y funcionan de igual manera que una calculadora, de un lado se ingresan los datos, esta realiza una operación y muestra el resultado, como lo muestra la siguiente figura.



**Figura 7.** Diagrama de Bloques Compuerta

Cada una de las compuertas lógicas se representa mediante un Símbolo, y la operación que realiza (Operación lógica) se corresponde con una tabla, llamada Tabla de Verdad.

#### **Operación Inversión o Negación (complemento)) Compuerta Not**

Esta operación se indica con una barra sobre la variable o por medio de un apóstrofe en el lado superior derecho de la variable, en este curso emplearemos esta última notación. El apóstrofe (') es un operador algebraico que invierte el valor de una variable, es decir, si  $X$  denota la señal de entrada de un inversor, entonces  $X'$  representa el complemento de tal señal.

*Ejemplo* Si  $X = a = 0$  entonces  $X' = s = 1$ .



**Figura 8.** Símbolo y Tabla de Verdad Compuerta Not

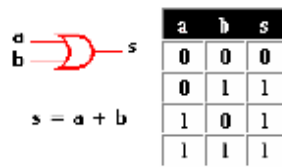
#### **Suma Booleana - Compuerta OR**

La Compuerta OR tiene dos entradas como mínimo y su operación lógica es una suma Booleana representada por medio un signo más entre las dos variables de entrada.

La suma Booleana de las variables  $a$  y  $b$  se enuncia de la siguiente forma,  $s = a + b$

La suma Booleana es  $1$  si alguna de las variables lógicas de la suma es  $1$  y es  $0$  cuando todas las variables son  $0$ . Esta operación se asimila a la conexión paralela de contactos.

Bueno, todo va bien hasta que  $1 + 1 = 1$ , el tema es que se trata de una compuerta O Inclusiva es como a y/o b, es decir, basta que una de ellas sea  $1$  para que su salida sea también  $1$ .



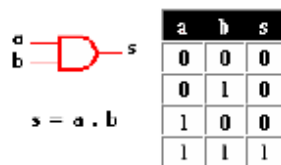
**Figura 9.** Símbolo y Tabla de Verdad Compuerta OR

### **Multiplicación Booleana – Compuerta AND**

La Compuerta AND tiene dos entradas como mínimo y su operación lógica es una multiplicación booleana representada por medio un signo ( $\cdot$ ) entre las dos variables de entrada. No es un producto aritmético, aunque en este caso coincidan. Observa que su salida será alta si sus dos entradas están a nivel alto.

La multiplicación booleana de las variables  $A$  y  $B$  se enuncia de la siguiente forma,  $X = A \cdot B$

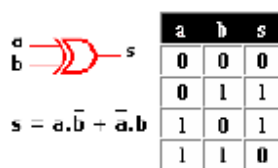
La multiplicación booleana es 1 si todas las variables lógicas son 1, pero si alguna es 0, el resultado es 0. La multiplicación booleana se asimila a la conexión serie de contactos.



**Figura 10.** Símbolo y Tabla de Verdad Compuerta AND

### **Compuerta EXOR o XOR**

Es OR Exclusiva en este caso con dos entradas y lo que hará con ellas será una suma lógica entre  $a$  por  $b$  invertida y  $a$  invertida por  $b$ . Al ser O Exclusiva su salida será 1 si una y sólo una de sus entradas es 1.



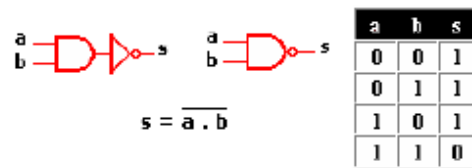
**Figura 11.** Símbolo y Tabla de Verdad Compuerta EXOR

### **b. Compuertas Lógicas Combinadas**

Al agregar una compuerta NOT a cada una de las compuertas anteriores los resultados de sus respectivas tablas de verdad se invierten, y dan origen a tres nuevas compuertas llamadas NAND, NOR y NOR-EX... Veamos ahora como son y cual es el símbolo que las representa...

#### **Compuerta NAND**

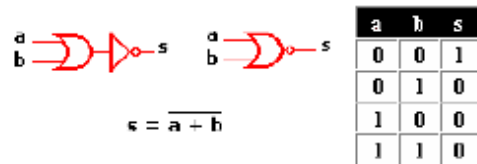
Responde a la inversión del producto lógico de sus entradas, en su representación simbólica se reemplaza la compuerta NOT por un círculo a la salida de la compuerta AND.



**Figura 12.** Símbolo y Tabla de Verdad Compuerta NAND

### Compuerta NOR

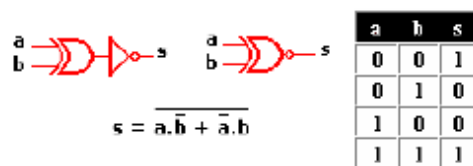
El resultado que se obtiene a la salida de esta compuerta resulta de la inversión de la operación lógica o inclusiva es como un no a y/o b. Igual que antes, solo agregar un círculo a la compuerta OR y ya se tiene una NOR.



**Figura 13.** Símbolo y Tabla de Verdad Compuerta NOR

### Compuerta NEXOR

Es simplemente la inversión de la compuerta EXOR, los resultados se pueden apreciar en la tabla de verdad y el símbolo que la representa en la figura 14, que bien se podría comparar con la figura 13 (anterior) y notar la diferencia.



**Figura 14.** Símbolo y Tabla de Verdad Compuerta NEXOR

### Compuerta YES o Buffer

En realidad no realiza ninguna operación lógica, su finalidad es amplificar un poco la señal (o refrescarla si se puede decir). Como se puede ver en la figura 15 la señal de salida es la misma que de entrada.



**Figura 15.** Símbolo y Tabla de Verdad Compuerta YES o Buffer**c. Circuitos Integrados y Circuito de Prueba**

Existen varias familias de Circuitos integrados, pero las más comunes y empleadas en Colombia son los TTL y CMOS. Estos Integrados se pueden caracterizar por el número que corresponde a cada familia según su composición.

Por ejemplo; Los TTL se corresponden con la serie 5400, 7400, 74LSXX, 74HCXX, 74HCTXX,... algunos 3000 y 9000.

Los C-MOS y MOS se corresponde con la serie CD4000, CD4500, MC14000, 54C00 ó 74C00, ...

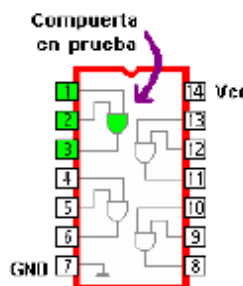
La pregunta de rigor... **¿Cual es la diferencia entre uno y otro?**, veamos...

Los C-MOS, el máximo nivel de tensión que soportan llega en algunos casos a +15V, mientras que para los TTL el nivel superior de tensión alcanza en algunos casos a los +12V aproximadamente, pero claro estos son límites extremos, lo común en estos últimos es utilizar +5V.

Otra característica es la velocidad de transmisión de datos, resulta ser, que los circuitos TTL son más rápidos (Comunicación Paralela) que los C-MOS (Comunicación Serial), por eso su mayor uso en sistemas de computación.

Es importante buscar la hoja de datos o datasheet del integrado que necesite, distribuido de forma gratuita por cada fabricante y disponible en Internet, o en el Manual ECG o NTE semiconductores.

Ejemplo Caso Circuito Integrado 74LS08, un TTL, es una cuádruple compuerta AND. Es importante notar el sentido en que están numerados los pines, lo cual es general para todo tipo de integrado.

**Figura 16.** Hoja de Datos Circuito Integrado 74LS08

Comenzaremos con este integrado para verificar el comportamiento de las compuertas vistas anteriormente. El representado en la figura 17 marca una de las compuertas que será puesta a prueba, para ello utilizaremos un fuente regulada de +5V, un LED una resistencia de 220  $\Omega$  (ohm), y por supuesto el IC que corresponda y la placa de prueba.





**Figura 17.** Prueba de una de las compuertas del Circuito Integrado 74LS08

En el esquema está marcada la compuerta, como 1 de 4 disponibles en el Integrado 74LS08, los extremos a y b son las entradas que se deberán llevar a un 1 lógico (+5V) ó 0 lógico (GND), el resultado en la salida s de la compuerta se verá reflejado en el LED, LED encendido (1 lógico) y LED apagado (0 lógico), no olvide conectar los terminales de alimentación que en este caso son el pin 7 a GND y el 14 a +5V. Montado en la placa de prueba te quedaría algo así en el Protoboard de la figura 18.



**Figura 18.** Esquema en Protoboard Prueba de una de las compuertas del Circuito Integrado 74LS08

Esto es a modo de ejemplo, Sólo debes reemplazar IC1, que es el Circuito Integrado que está a prueba para verificar su tabla de verdad.

#### d. Propiedades de las Operaciones Booleanas

Las operaciones booleanas están regidas por tres leyes similares a las del álgebra convencional. Estas incluyen las leyes conmutativas de la suma y la multiplicación y la ley distributiva.

##### **Leyes conmutativas en dos variables**

1. Ley conmutativa de la suma se enuncia como sigue

$$X + Y = Y + X$$

En aplicación a los circuitos digitales, podríamos decir que no importa el orden de conexión de las entradas a una compuerta *OR*.

2. Ley conmutativa de la multiplicación

$$X \cdot Y = Y \cdot X$$

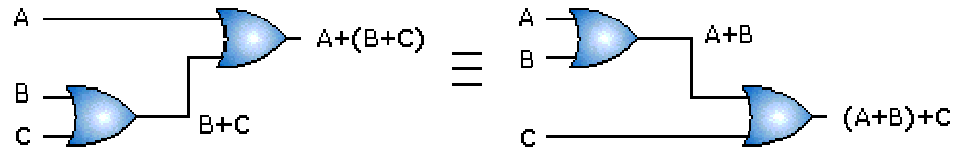
En aplicación a los circuitos digitales, podríamos decir que no importa el orden de conexión de las entradas a una compuerta *AND*.

##### **Leyes asociativas en tres variables**

1. Ley asociativa de la adición, se escribe en forma algebraica de la siguiente forma

$$A + (B + C) = (A + B) + C$$

En la figura 19 se muestra la aplicación de la propiedad a las compuertas OR,

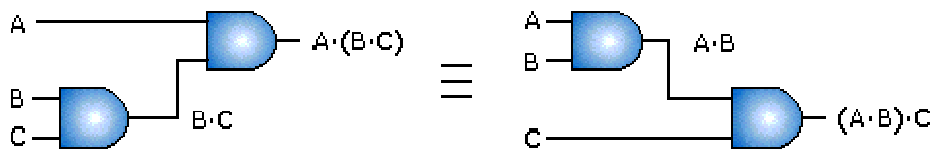


**Figura 19.** Ley asociativa de la adición

## 2. Ley asociativa de la multiplicación

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

En la figura 20 se muestra la aplicación de la propiedad a las compuertas AND,



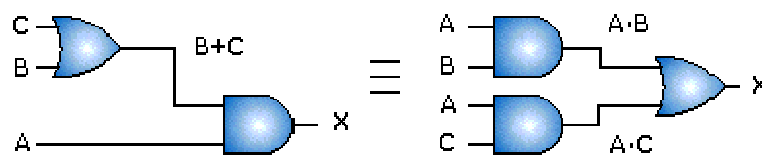
**Figura 20.** Ley asociativa de la multiplicación

### **Ley distributiva para tres variables**

En el álgebra de Boole, la multiplicación lógica se distribuye sobre la suma lógica,

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

En la figura 21 se muestra la aplicación de la propiedad a las compuertas AND y OR,



**Figura 21.** Ley distributiva para tres variables

**e. Teoremas Booleanos**

Los teoremas booleanos son enunciados siempre verdaderos, lo que permite la manipulación de expresiones algebraicas, facilitando el análisis ó síntesis de los circuitos digitales. Los teoremas booleanos son los siguientes:

1.  $X + 0 = X$
2.  $X + 1 = 1$
3.  $X \cdot 0 = 0$
4.  $X \cdot 1 = X$
5.  $(X')' = X$
6.  $X + X = X$
7.  $X \cdot X = X$
8.  $X + X' = 1$
9.  $X \cdot X' = 0$
10.  $X + XY = X$
11.  $X + X' \cdot Y = X + Y$
12.  $X \cdot Y + X \cdot Y' = X$  (*Teorema de combinación*)
13.  $(X + Y)(X + Y') = X + X \cdot Y' + X \cdot Y = X$
14.  $X \cdot Y + X \cdot Z + Y \cdot Z' = XZ + Y \cdot Z'$  (*Consenso*)

El teorema 12 se conoce como la ley distributiva para tres variables.

*Demostración teorema 12:*

$$X \cdot Y + X \cdot Y' = X$$

Utilizando la ley distributiva para tres variables

$$X \cdot Y + X \cdot Y' = X \cdot (Y + Y')$$

Aplicando el teorema 8 se tiene,

$$X \cdot Y + X \cdot Y' = X \cdot 1$$

Dando como resultado,

$$X \cdot Y + X \cdot Y' = X$$

Esta expresión indica que la suma de dos productos canónicos adyacentes, es decir que difieren en una sola de las variables, se reduce al producto de los demás términos suprimiéndose dicha variable. El teorema 13 es otro caso del teorema de combinación. Los teoremas 12 y 13 se utilizarán en las lecciones siguientes de forma sistemática para sintetizar circuitos lógicos con los métodos de mapas de karnaugh y el algoritmo de Quine-McCluskey.

**f. Teoremas de DeMorgan**

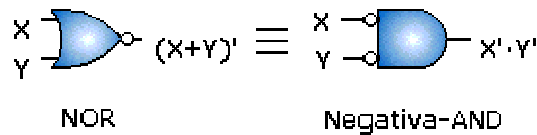
Los teoremas de DeMorgan demuestran la equivalencia entre las puertas *NAND* y negativa - *OR*, y las puertas *NOR* y negativa – *AND*.

1. El complemento de la suma de variables es igual al producto de los complementos de las variables.

$$(X_1 + X_2 + \dots + X_n)' = X_1' \cdot X_2' \cdot \dots \cdot X_n'$$

En el caso de dos variables se tiene,  $(X + Y)' = X' \cdot Y'$

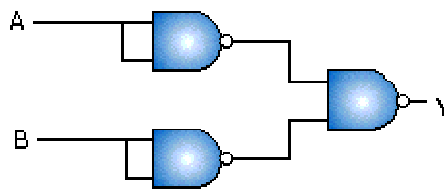
El circuito equivalente a la ecuación anterior se muestra en la figura 22.



**Figura 22.** Símbolo lógico para la compuerta NOR.

*Ejemplo:* Obtener una compuerta OR utilizando compuertas NAND.

$$Y = (A + B) = [(A + B)']' = (A' \cdot B')'$$



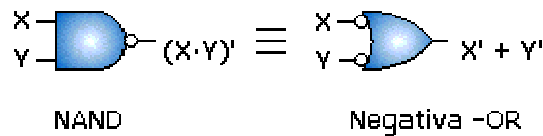
**Figura 23.** Compuerta OR utilizando compuertas NAND

2. El complemento del producto de variables es igual a la suma de los complementos de las variables.

$$(X_1 \cdot X_2 \cdot \dots \cdot X_n)' = X_1' + X_2' + \dots + X_n'$$

En el caso de dos variables se tiene,  $(X \cdot Y)' = X' + Y'$

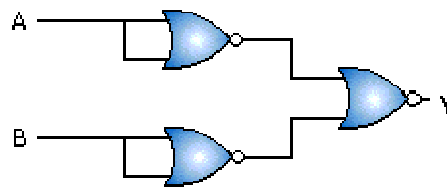
El circuito equivalente en dos variables a la ecuación se muestra en la figura 24.



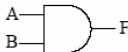
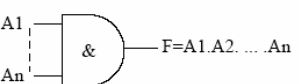

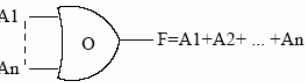
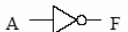
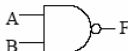
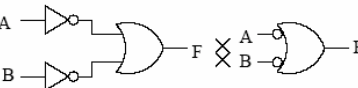

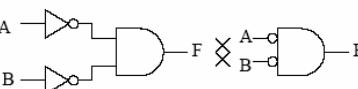

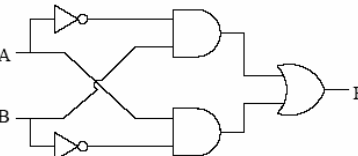

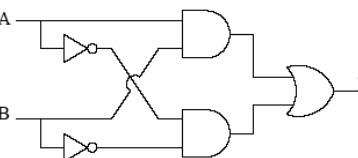
**Figura 24.** Símbolo lógico para la compuerta NOR.

*Ejemplo:* Obtener una compuerta AND utilizando compuertas NOR.

$$Y = A \cdot B = [(A \cdot B)']' = (A' + B')'$$



**Figura 25.** Circuito lógico para la compuerta AND

	SIMBOLO	FUNCION	TABLA	DATOS DE INTERES															
AND		$F = A \cdot B$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1	
A	B	F																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OR		$F = B + A$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1	
A	B	F																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
NOT		$F = \bar{A}$	<table><tr><th>A</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	F	0	1	1	0										
A	F																		
0	1																		
1	0																		
NAND		$F = \overline{A \cdot B} = \bar{A} + \bar{B}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0	
A	B	F																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOR		$F = \overline{A + B} = \bar{A} \cdot \bar{B}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0	
A	B	F																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	
XOR	 OR EXCLUSIVA	$F = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0	
A	B	F																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
NXOR	 NOR EXCLUSIVA	$F = \overline{A \oplus B} = \bar{A} \cdot B + A \cdot \bar{B}$ $= \overline{\bar{A} \cdot B + A \cdot \bar{B}} = (A + \bar{B})(\bar{A} + B) = A \cdot B + \bar{B} \cdot \bar{A}$	<table><tr><th>A</th><th>B</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	1	
A	B	F																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

**Tabla 8.1.** Propiedades, Tablas y Representación Gráfica del Algebra de Boole

### 2.3. Simplificación de Expresiones Lógicas

El objetivo de la simplificación de expresiones lógicas es reducir la expresión al menor número posible de términos. Las expresiones lógicas se pueden simplificar utilizando los teoremas anteriores.

*Ejemplo:*

$$1. F = A \cdot B' \cdot C + A \cdot B' C';$$

$$F = A \cdot B' \cdot (C + C');$$

$$F = A \cdot B'$$

$$2. F = (A' + B) \cdot (A + B');$$

$$F = A \cdot A' + A' \cdot B' + A \cdot B + B \cdot B';$$

$$F = A' \cdot B' + A \cdot B$$

$$3. F = [(A' + C) \cdot (B + D')]';$$

$$F = (A' + C)' + (B + D')';$$

$$F = A \cdot C' + B' \cdot D$$

$$4. F = (X + Z') \cdot (Z + W \cdot Y)' + (V \cdot Z + W \cdot X') \cdot (Y + Z)';$$

$$F = (X + Z') \cdot [Z' \cdot (W' + Y')] + [(V \cdot Z + W \cdot X') \cdot (Y' \cdot Z')]$$

$$F = (X + Z') \cdot (Z' \cdot W' + Z' \cdot Y') + V \cdot Y' \cdot Z \cdot Z' + W \cdot X' \cdot Y' \cdot Z'$$

$$F = W' \cdot X \cdot Z' + X \cdot Y' \cdot Z' + Z' \cdot Z' \cdot W' + Z' \cdot Z' \cdot Y' + W \cdot X' \cdot Y' \cdot Z'$$

$$F = W' \cdot X \cdot Z' + X \cdot Y' \cdot Z' + W' \cdot Z' + Y' \cdot Z' + W \cdot X' \cdot Y' \cdot Z'$$

$$F = W' \cdot Z' \cdot (1 + X) + Y' \cdot Z' \cdot (1 + X) + W \cdot X' \cdot Y' \cdot Z'$$

$$F = W' \cdot Z' + Y' \cdot Z' + W \cdot X' \cdot Y' \cdot Z'$$

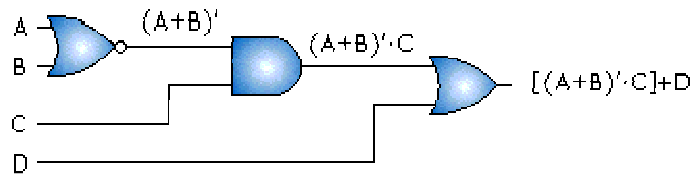
$$F = W' \cdot Z' + Y' \cdot Z' \cdot (1 + W \cdot X')$$

$$F = Z' \cdot (W' + Y')$$

### 2.4. Implementación de Funciones Lógicas mediante Compuertas.

La forma más fácil de encontrar la expresión de un circuito lógico consiste en comenzar con las entradas situadas más a la izquierda e ir avanzando hasta la salida de cada compuerta lógica, obteniendo la expresión para cada una de ellas. Al final del recorrido se debe tener la expresión para todo el circuito. La expresión resultante podemos simplificarla para obtener una más sencilla y así obtener un circuito más reducido.

*Ejemplo* Encontrar la expresión para el circuito de la figura 26.



**Figura 26.** Símbolo lógico para la compuerta NOR.

1. La expresión de la compuerta NOR situada a la izquierda cuyas entradas son A y B es  $(A+B)'$ . Esta es la primera entrada de la compuerta AND situada a la derecha.
2. La expresión de la compuerta AND cuyas entradas son  $(A+B)'$  y C es  $(A+B)' \cdot C$ .
3. La salida de la compuerta AND es la primera entrada de la compuerta OR del extremo derecho. Por lo tanto, la expresión de esta compuerta OR es  $[(A+B)' \cdot C] + D$ .

## 2.5. Síntesis de Diseño de Circuitos Combinacionales

Síntesis se entiende como la obtención de circuitos lógicos, a partir de una descripción inicial que utiliza el lenguaje convencional y luego es transferida a una tabla de verdad.

Una tabla de verdad es una representación básica de una función lógica, en la cual se listan las salidas del circuito lógico para las posibles combinaciones de entrada. Las combinaciones de entrada están ordenadas por renglones (líneas) y cada renglón contiene su salida respectiva. Por ejemplo, la tabla de verdad para una función lógica de 3 variables, tendrá 8 líneas para 8 combinaciones de entrada, conteniendo cada línea, su salida respectiva. En la tabla 9. Se ilustra una función de 3 variables para el caso mencionado.

Renglón o línea	A	B	C	Función de salida	Mintérmino	Maxtérmino
0	0	0	0	$F(0,0,0)$	$A' \cdot B' \cdot C'$	$A+B+C$
1	0	0	1	$F(0,0,1)$	$A' \cdot B' \cdot C$	$A+B+C'$
2	0	1	0	$F(0,1,0)$	$A' \cdot B \cdot C'$	$A+B'+C$
3	0	1	1	$F(0,1,1)$	$A' \cdot B \cdot C$	$A+B'+C'$
4	1	0	0	$F(1,0,0)$	$A \cdot B' \cdot C'$	$A'+B+C$
5	1	0	1	$F(1,0,1)$	$A \cdot B' \cdot C$	$A'+B+C'$
6	1	1	0	$F(1,1,0)$	$A \cdot B \cdot C'$	$A'+B'+C$
7	1	1	1	$F(1,1,1)$	$A \cdot B \cdot C$	$A'+B'+C'$

**Tabla 9.** Funciones de salida, Maxtérminos y Mintérminos

## 2.6. Métodos para Sintetizar Circuitos Lógicos

Los métodos para sintetizar circuitos lógicos requieren en primer lugar, la comprensión de algunos conceptos, entre ellos:

- **Literal:** Variable o el complemento de una variable.

Ejemplo:  $X'$ ,  $Y'$ ,  $X$ ,  $Y$ .

- **Dominio de una expresión booleana:** Es el conjunto de variables contenido en una expresión booleana.

Ejemplo: Determine el dominio de la expresión  $X' \cdot Y \cdot Z + X \cdot Y' \cdot Z \cdot W$ .

El dominio es  $X$ ,  $Y$ ,  $Z$ ,  $W$ .

- **Término normal:** Un producto o término suma en donde ninguna variable aparece repetida.

Ejemplo de término repetido:  $X \cdot Y \cdot Y$ ,  $Z \cdot X' \cdot X' \cdot Y$

Ejemplo de término no repetido:  $X' \cdot Y \cdot Z$ ,  $Z \cdot Y' \cdot X$

- **Término producto:** Un solo literal o el producto lógico (multiplicación booleana) de dos o más literales.

Ejemplo:  $X'$ ,  $X \cdot Y'$ ,  $Z \cdot Y$ ,  $X \cdot Y' \cdot Z$

Un término producto es 1 sólo para una combinación de valores de las variables.

Ejemplo: El término producto  $X \cdot Y' \cdot Z$  es 1 sólo para  $X=1$ ,  $Y=0$  y  $Z=1$  y es 0 para el resto de combinaciones. El valor en binario será 101 ó 5 en decimal.

- **Término suma:** Un solo literal o una suma lógica (suma booleana) de dos o más literales.

Ejemplo:  $X$ ,  $X + Y'$ ,  $X' + Z'$ ,  $X + Y + Z$ ,  $X + Y' + Z'$

Un término suma es 1 cuando cualquier literal que lo compone es 1.

Ejemplo: El término  $X + Y' + Z'$  es 0 para  $X=0$  ó  $Y=1$  ó  $Z=1$  y es 1 para el resto de combinaciones. El valor en binario será 011 ó 3 en decimal.

- **Suma de productos:** Suma lógica de términos productos (Ver tabla 9).

Ejemplo:  $X' + X \cdot Y' + Z \cdot Y + X \cdot Y' \cdot Z$

**Forma estándar de la suma de productos:** Una suma de productos no se encuentra en su forma estándar cuando alguno de los términos producto no contiene alguna de las variables del dominio de la expresión.

Ejemplo



$X' \cdot Y \cdot Z + X \cdot Y' \cdot Z \cdot W$ . El dominio es  $X, Y, Z, W$ . El primer término producto no contiene el literal  $W$  ó  $W'$ .

Ejemplo

$X' \cdot Y \cdot Z' \cdot W + X \cdot Y \cdot Z \cdot W$ . En cada uno de los términos de la expresión aparecen todas las variables del dominio. Por lo tanto, la suma de productos está en su forma estándar.

- **Producto de sumas:** Producto lógico de términos suma (Ver tabla 9).

Ejemplo:  $X \cdot (X+Y') \cdot (X'+Z') \cdot (X+Y+Z) \cdot (X+Y'+Z')$ .

**Forma estándar del producto de sumas:** Un producto de sumas no se encuentra en su forma estándar cuando alguno de los términos suma no contiene alguna de las variables del dominio de la expresión.

Ejemplo

$(X'+W+Z') \cdot (X'+Y'+Z+W') \cdot (X+Y)$ . El dominio es  $X, Y, Z, W$ . El primer término suma no contiene el literal  $Y$  ó  $Y'$ . El tercer término suma no contiene los literales  $Z$  ó  $Z'$  y  $W$  ó  $W'$ .

Ejemplo

$(X' \cdot Y \cdot Z' \cdot W) \cdot (X \cdot Y' \cdot Z \cdot W)$ . En cada uno de los términos de la expresión aparecen todas las variables del dominio. Por lo tanto, el producto de sumas está en su forma estándar.

- **Mintérmino:** Es un término de producto con  $n$  literales en el cual hay  $n$  variables. De  $n$  variables obtenemos  $2^n$  mintérminos.

Ejemplo de mintérminos de 3 variables:  $X' \cdot Y' \cdot Z'$ ,  $X' \cdot Y' \cdot Z$ ,  $X' \cdot Y \cdot Z'$ ,  $X' \cdot Y \cdot Z$ ,  $X \cdot Y' \cdot Z'$ ,  $X \cdot Y' \cdot Z$ ,  $X \cdot Y \cdot Z'$ ,  $X \cdot Y \cdot Z$ . (Ver tabla 9.).

- **Maxtérmino:** Es un término de suma con  $n$  literales en el cual hay  $n$  variables. De  $n$  variables obtenemos  $2^n$  maxtérminos. (Ver tabla 9.).

Ejemplo de maxtérminos de 3 variables:  $X+Y+Z$ ,  $X+Y+Z'$ ,  $X+Y'+Z$ ,  $X+Y'+Z'$ ,  $X'+Y+Z$ ,  $X'+Y+Z'$ ,  $X'+Y'+Z$ ,  $X'+Y'+Z'$ . (Ver tabla 2.2.1.).

Los métodos existentes para sintetizar circuitos lógicos son:

- Suma de productos (*SDP*).
- Producto de sumas (*PDS*).
- Mapas de Karnaugh.
- Algoritmo de Quine – McCluskey.

**a. Método de Suma de Productos (SDP)**

La suma de productos de una función lógica es la suma de los minterminos correspondientes a las líneas de la tabla de verdad para las que la función produce una salida igual a 1. La función obtenida es la suma de productos.

*Ejemplo* Obtener la suma de productos para la función lógica de la tabla 10.

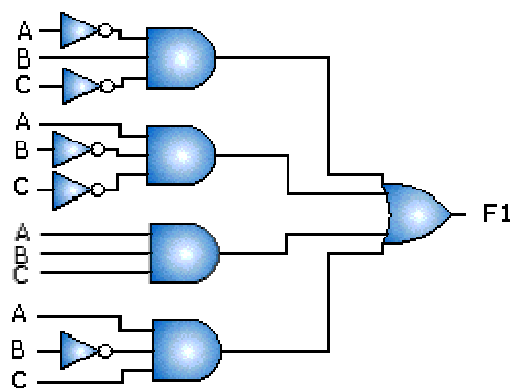
Línea	A	B	C	Función de salida $F_1$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

**Tabla 10.** Tabla de verdad para la función lógica  $F_1$

La función puede ser expresada conformando un término mínimo por cada combinación de variables que producen un 1 en la función para luego obtener la suma de todos los términos. La función lógica para la tabla 2.3.1 se determina expresando las combinaciones 010, 100, 101 y 111 como  $A'B \cdot C'$ ,  $A \cdot B' \cdot C'$ ,  $A \cdot B' \cdot C$  y  $A \cdot B \cdot C$ :

$$F_1 = \sum_{A,B,C} (2,4,5,7) = A' \cdot B \cdot C' + A \cdot B' \cdot C' + A \cdot B' \cdot C + A \cdot B \cdot C.$$

Cada mintermino de la función anterior representa una compuerta *AND* de tres entradas y la implementación de la función es posible a través de la aplicación de la operación *OR* a las salidas de las cuatro compuertas *AND*. Por tanto, el número total de compuertas *AND* dependerá del total de minterminos de la expresión. El circuito se muestra en la figura 27.



**Figura 27.** Circuito lógico para la función lógica  $F_1$ .

En una suma de productos se cumple la igualdad de la función al valor lógico 1 si al menos uno de sus términos productos es igual a 1.

*Ejemplo* Obtener la suma de productos para la función lógica de la tabla 11.

A	B	F <sub>2</sub>
0	0	0
0	1	1
1	0	1
1	1	0

**Tabla 11.** Tabla de verdad de la función F<sub>2</sub>.

En la tabla de verdad existen dos condiciones para las cuales la salida es 1. Estas son las siguientes:

1. La primera se presenta cuando A es Bajo(0) y B es Alto(1). El resultado 1 de esta condición se puede expresar como el producto lógico:

$$A' \cdot B$$

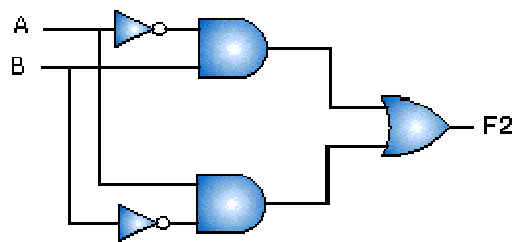
2. La segunda condición se presenta cuando A es 1 y B es 0. Esta condición ocasiona un resultado 1, si el producto lógico es:

$$A \cdot B'$$

Como cualquiera de estas dos (2) condiciones hace que la salida sea 1, entonces la función lógica que los representa es la suma lógica de los productos anteriores:

$$F_2 = A' \cdot B + A \cdot B' = A \oplus B$$

La representación de la función anterior con compuertas OR y AND se muestra en la figura 28.



**Figura 28.** Función F<sub>2</sub> utilizando compuertas AND Y OR

Esta función corresponde a la función *OR exclusiva*, cuya compuerta se representa en la figura 11.

*Ejemplo* Obtener la función SDP para la función lógica de la tabla 12. Simplificar la función y dibujarla.

A	B	F <sub>3</sub>
0	0	1
0	1	0
1	0	0
1	1	1

**Tabla 12.** Tabla de verdad de la función F<sub>3</sub>

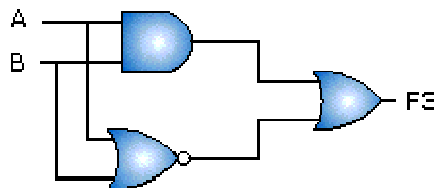
Utilizando suma de productos para las líneas 1 y 4 de la tabla se obtiene,

$$F_3 = A' \cdot B' + A \cdot B, \text{ simplificando}$$

$$F_3 = (A+B)' + A \cdot B$$

$$F_3 = (A \oplus B)'$$

El circuito lógico de la función anterior se muestra en la figura 29.



**Figura 29.** Función F<sub>3</sub> utilizando compuertas AND, NOR y OR.

El símbolo lógico de la compuerta *NEXOR* - Se muestra en la figura 14.

### **Conversión de una expresión lógica a formato de suma de productos**

La metodología empleada en la transformación de una suma de productos a su forma estándar se basa en el teorema 6, que establece que una variable sumada con su complemento es siempre igual a 1;  $A + A' = 1$ . Los pasos son los siguientes:

1. Los términos producto que no contengan la(s) variable(s) del dominio, multiplicarlos por un término formado por dicha variable más el complemento de la misma (teorema 6).
2. Repetir el paso 1 para todos los términos de la expresión que no contengan todas las variables (o sus complementos) del dominio. Resolver los términos intervenidos.

*Ejemplo* Convertir la expresión booleana  $A \cdot B \cdot C' + B \cdot C + A'$  a su forma estándar.

El dominio de la expresión es el conjunto de variables A, B y C. Se observa la falta de formato estándar para el segundo y tercer término producto. Sobre ellos se aplicará el procedimiento, para luego volver a agrupar toda la expresión:

Término  $B \cdot C$

$$B \cdot C = B \cdot C \cdot (A + A') = A \cdot B \cdot C + A' \cdot B \cdot C$$

Término  $A$

$A' = A' \cdot (C + C') = A' \cdot C + A' \cdot C'$ ; la expresión aún no tiene el formato estándar, entonces multiplicamos cada término por  $(B + B')$

$$A' \cdot C \cdot (B + B') + A' \cdot C' \cdot (B + B') = A' \cdot B \cdot C + A' \cdot B' \cdot C + A' \cdot B \cdot C' + A' \cdot B' \cdot C'$$

La expresión en su formato estándar es:

$$A \cdot B \cdot C' + B \cdot C + A' = A \cdot B \cdot C + A' \cdot B \cdot C + A' \cdot B \cdot C' + A' \cdot B' \cdot C + A' \cdot B \cdot C' + A' \cdot B' \cdot C'$$

### b. Método de producto de sumas (PDS)

El producto de sumas de una función lógica es la multiplicación de los maxtérminos correspondientes a las líneas de la tabla de verdad para las que la función produce una salida igual a 0. La función obtenida es el producto de sumas.

Ejemplo Obtener el producto de sumas para la función lógica de la tabla 13.

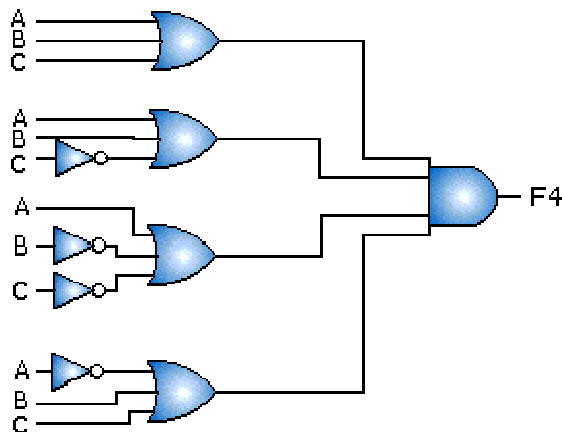
Renglón o línea	A	B	C	Función de salida $F_4$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

**Tabla 13.** Tabla de verdad para la función lógica  $F_4$

La función puede ser expresada conformando un término máximo para cada combinación de variables que producen un 0 en la función y luego obtener el producto de todos los términos. La función lógica para la tabla 2.3.4 se determina expresando las combinaciones 000, 001, 011 y 110 como  $(A+B+C)$ ,  $(A+B+C')$ ,  $(A+B'+C)$  y  $(A'+B+C)$ . La función lógica es la siguiente:

$$F_4 = \prod_{A,B,C} (0, 1, 3, 4) = (A+B+C) \cdot (A+B+C') \cdot (A+B'+C) \cdot (A'+B+C).$$

Cada maxtérmino de la función anterior representa una compuerta OR de tres entradas y la implementación de la función es posible a través de la aplicación de la operación AND a las salidas de las cuatro compuertas AND. Por tanto, el número total de compuertas AND dependerá del total de mintérminos de la expresión. El circuito se muestra en la figura 30.



**Figura 30.** Circuito lógico para la función lógica  $F_4$

Un producto de sumas es igual a 0 si al menos uno de los términos suma es igual a 0.

Ejemplo Obtener el producto de sumas para la función lógica de la tabla 14.

A	B	$F_5$
0	0	0
0	1	1
1	0	1
1	1	0

**Tabla 14.** Tabla de verdad de la función EXOR

Considere el complemento de la función de Boole  $F_5$ . Este puede obtenerse de la tabla 14. Formando un término mínimo por cada combinación que produce un cero y luego haciendo la suma de los términos. El complemento de  $F_5$  se expresa así:

$$F_5' = A' \cdot B' + A \cdot B$$

La expresión  $F_5$  se obtiene la negar  $F_5'$ :

$$F_5 = (F_5')' = (A' \cdot B' + A \cdot B)' = (A' \cdot B')' \cdot (A \cdot B)' = [(A')' + (B')'] \cdot (A' + B') = (A + B) \cdot (A' + B')$$

Si cualquiera de los términos del *PDS* es cero, la función es cero.

De los 2 métodos anteriores, se pueden escoger algunos criterios para aplicar un método u otro, siendo estos los siguientes:

- Si en la última columna de la tabla de verdad, o sea en la columna que indica los resultados, sí predominan los ceros es más conveniente utilizar las suma de productos.
- Si en la columna que indica los resultados, predominan los unos, es más conveniente utilizar el método del producto de sumas.

### c. Mapas de Karnaugh

Un mapa de Karnaugh es una representación gráfica de una función lógica a partir de una tabla de verdad. El número de celdas del mapa es igual al número de combinaciones que se pueden obtener con las variables de entrada.

El Mapa de Karnaugh representa la misma tabla de verdad a través de una matriz, en la cual, en la primera fila y la primera columna se indican las posibles combinaciones de las variables. Los mapas se pueden utilizar para 2, 3, 4 y 5 variables.

Para 2 Variables			Para 3 Variables					Para 4 Variables				
a\b	0	1	a\b c	00	01	11	10	ab\cd	00	01	11	10
0	0	1	0	0	1	3	2	00	0	1	3	2
1	2	3	1	4	5	7	6	01	4	5	7	6
								11	12	13	15	14
								10	8	9	11	10

**Figura 31.** Mapa de Karnaugh para 2, 3 y 4 variables.

Analicemos el mapa para cuatro variables, las dos primeras columnas (columnas adyacentes) difieren sólo en la variable d, y c permanece sin cambio, en la segunda y tercera columna (columnas adyacentes) cambia c, y d permanece sin cambio, ocurre lo mismo en las filas. En general se dice que...

**Dos columnas o filas adyacentes sólo pueden diferir en el estado de una de sus variables**

Observa también que según lo dicho anteriormente la primer columna con la última serían adyacentes, al igual que la primer fila y la última, ya que, sólo difieren en una de sus variables.

### Mapa de Karnaugh empleando Suma de Productos (SDP)

La simplificación de expresiones lógicas mediante el mapa de Karnaugh utiliza un método gráfico basado en la Suma de Productos.

### Mapa de Karnaugh de tres variables

El mapa de Karnaugh se construye a partir de la tabla de verdad de la función lógica. El mapa por medio de una matriz de 8 celdas, representa los ocho mintérminos posibles que se pueden obtener con tres variables, en un arreglo de una matriz de 2x4. Por tanto, la primera fila contiene el primer valor posible ("0") y la segunda fila el valor ("1").

Las variables 2 y 3 se agrupan por columna y se distribuyen en las cuatro columnas de acuerdo a las combinaciones posibles para obtener los mintérminos requeridos. Sus valores son 00, 01, 10 y 11. Por ejemplo, la celda  $m_2$  corresponde al mintérmino 2, ubicado en la fila 0 y la columna 10. La unión de estos dos números da el número 010, cuyo equivalente es el término  $A' \cdot B \cdot C'$  ó el decimal 2. La tabla 15. Muestra el mapa de Karnaugh para 3 variables.

Línea	A	B	C	Mintérmino	Mintérmino	Función de
-------	---	---	---	------------	------------	------------

					$m_x$	Salida
0	0	0	0	$A' \cdot B' \cdot C'$	$m_0$	$F(0,0,0)$
1	0	0	1	$A' \cdot B' \cdot C$	$m_1$	$F(0,0,1)$
2	0	1	0	$A' \cdot B \cdot C'$	$m_2$	$F(0,1,0)$
3	0	1	1	$A' \cdot B \cdot C$	$m_3$	$F(0,1,1)$
4	1	0	0	$A \cdot B' \cdot C'$	$m_4$	$F(1,0,0)$
5	1	0	1	$A \cdot B' \cdot C$	$m_5$	$F(1,0,1)$
6	1	1	0	$A \cdot B \cdot C'$	$m_6$	$F(1,1,0)$
7	1	1	1	$A \cdot B \cdot C$	$m_7$	$F(1,1,1)$

(a)

		BC			
		$B'C'$	$B'C$	$BC$	$BC'$
A	$A'$	$m_0$	$m_1$	$m_3$	$m_2$
	A	$m_4$	$m_5$	$m_7$	$m_6$

(b)

		BC			
		00	01	11	10
A	0	$A'B'C'$	$A'B'C$	$A'BC$	$A'BC'$
	1	$AB'C'$	$AB'C$	$ABC$	$ABC'$

(c)

Tabla 15. Mapa de tres variables

La característica de ordenamiento de un mapa de Karnaugh radica en el cambio de un solo BIT en los términos de las celdas adyacentes de filas y columnas. En la tabla 15. Las entradas  $BC$  se colocan secuencialmente, cambiando cada vez una sola variable, por eso resulta el orden: 00, 01, 11 y 10. En la interactividad 2.4.1., la pulsación de cada cuadro activa el mintérmino correspondiente.

Por ejemplo, la variable  $C$  está negada en  $m_4$  y  $m_5$  no lo está, mientras que  $A$  y  $B$  no cambia. Las celdas de los bordes superior e inferior e izquierdo y derecho también cumplen esta condición al agruparlas unas a otras. En el teorema 12 de la lección 1, se demuestra que la suma de los términos mínimos en celdas adyacentes pueden ser simplificadas en un término AND de dos literales. Por consiguiente, aplicando el teorema para los términos  $m_4$  y  $m_5$  del mapa se tiene:

$$m_4 + m_5 = A \cdot B' \cdot C' + A \cdot B' \cdot C = A \cdot B' \cdot (C' + C) = A \cdot B$$

Los términos  $m_4$  y  $m_6$  se pueden asociar de la misma forma:

$$m_4 + m_6 = A \cdot B' \cdot C' + A \cdot B \cdot C' = A \cdot C' \cdot (B' + B) = A \cdot C'$$

*Ejemplo* Simplificar la función  $F_1 = \sum (m_3, m_4, m_5, m_6, m_7)$ .

$$F_1 = \sum (m_3, m_4, m_5, m_6, m_7) = A' \cdot B \cdot C + A \cdot B' \cdot C' + A \cdot B' \cdot C + A \cdot B \cdot C' + A \cdot B \cdot C$$



Aplicando el teorema 6 de la lección 1 para el término  $A \cdot B \cdot C$ .

$$F_1 = \sum (m_3, m_4, m_5, m_6, m_7) = \sum (m_4, m_5, m_6, m_7) + \sum (m_3, m_7) = [A \cdot B' \cdot C' + A \cdot B' \cdot C + A \cdot B \cdot C' + A \cdot B \cdot C] + [A' \cdot B \cdot C + A \cdot B \cdot C].$$

El primer término en la sumatoria es el grupo 1 y el segundo término corresponde al grupo 2. En un mapa de karnaugh, los mintérminos de cada grupo se relacionarían a través de lazos independientes.

Desarrollando la expresión,

$$F_1 = [A \cdot B' \cdot (C' + C) + A \cdot B \cdot (C' + C)] + [B \cdot C \cdot (A' + A)] = A \cdot B' \cdot (1) + A \cdot B \cdot (1) + B \cdot C \cdot (1) = A \cdot (B' + B) + B \cdot C = A + B \cdot C.$$

El mapa se construye colocando un 1 en las celdas correspondientes a los mintérminos presentes en la función de salida. Por ejemplo, para el término  $F(1,1,0) = A \cdot B \cdot C' = 1$  se situaría un 1 en la celda 110. Para los mintérminos no presentes en la función se pone un 0. Por ejemplo el término  $F(0,0,1) = A' \cdot B' \cdot C = 0$ , será una celda con valor 0 en la celda 001.

Después de situar los unos en el mapa, se procede con la agrupación de 1s, la determinación del término producto correspondiente a cada grupo y la suma de los términos producto obtenidos. La determinación del término producto se realiza de acuerdo los siguientes criterios:

1. Una celda representa un mintérmino, dando como resultado un término de cuatro literales.
2. Dos celdas agrupadas pueden representar la asociación de dos mintérminos, dando como resultado un término de dos literales.
3. Cuatro celdas agrupadas pueden representar la asociación de cuatro mintérminos, dando como resultado un término de un literal.
4. Ocho celdas agrupadas representan un valor de función igual a 1.

*Ejemplo* Sea la función del ejemplo anterior, simplificarla por medio del método del mapa.

La tabla de verdad del ejemplo anterior es la siguiente,

Línea	A	B	C	Salida F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

**Tabla 16.** Tabla de verdad de la función  $F_1$ .

El mapa de Karnaugh se configura de acuerdo a los minterminos iguales a 1 y las celdas se agrupan tal como en la figura 32

		BC			
		00	01	11	10
A	0	0	0	1	0
	1	1	1	1	1

**Figura 32.** Mapa de Karnaugh de la función  $F_1$ .

El primer grupo se forma con los minterminos  $m_4$ ,  $m_5$ ,  $m_6$  y  $m_7$  y el segundo grupo con los minterminos  $m_3$  y  $m_7$ .

Del primer grupo resulta el término  $A$ , ya que, para las cuatro columnas de la tabla existen transiciones entre las variables  $B$  y  $C$ . El segundo grupo da como resultado el término  $BC$  por el cambio existente en la variable  $A$ .

En total, la función queda reducida a la expresión:

$$F_1 = A + B \cdot C$$

### Mapa de Karnaugh de cuatro variables

La construcción de un mapa de Karnaugh de 4 variables es similar al de 3 variables. La diferencia radica en el número de variables de entrada. El mapa por medio de una matriz de 16 celdas, representa los 16 minterminos posibles ( $2^4$ ) que se pueden obtener con cuatro variables de entrada, en un arreglo de 4 x 4. La disposición de celdas en el mapa se muestra en la tabla 17.

Línea	A	B	C	D	Mintérmino	Mintérmino $m_x$	Función de Salida
0	0	0	0	0	$A'B'C'D'$	$m_0$	$F(0,0,0,0)$
1	0	0	0	1	$A'B'C'D$	$m_1$	$F(0,0,0,1)$
2	0	0	1	0	$A'B'C'D'$	$m_2$	$F(0,0,1,0)$
3	0	0	1	1	$A'B'C'D$	$m_3$	$F(0,0,1,1)$
4	0	1	0	0	$A'B'C'D'$	$m_4$	$F(0,1,0,0)$
5	0	1	0	1	$A'B'C'D$	$m_5$	$F(0,1,0,1)$
6	0	1	1	0	$A'B'C'D'$	$m_6$	$F(0,1,1,0)$
7	0	1	1	1	$A'B'C'D$	$m_7$	$F(0,1,1,1)$
8	1	0	0	0	$A'B'C'D'$	$m_8$	$F(1,0,0,0)$
9	1	0	0	1	$A'B'C'D$	$m_9$	$F(1,0,0,1)$
10	1	0	1	0	$A'B'C'D'$	$m_{10}$	$F(1,0,1,0)$
11	1	0	1	1	$A'B'C'D$	$m_{11}$	$F(1,0,1,1)$
12	1	1	0	0	$A'B'C'D'$	$m_{12}$	$F(1,1,0,0)$
13	1	1	0	1	$A'B'C'D$	$m_{13}$	$F(1,1,0,1)$
14	1	1	1	0	$A'B'C'D'$	$m_{14}$	$F(1,1,1,0)$
15	1	1	1	1	$A'B'C'D$	$m_{15}$	$F(1,1,1,1)$

(a)

AB \ CD	$C'D'$	$C'D$	$C'D$	$C'D'$
$A'B'$	$m_0$	$m_1$	$m_3$	$m_2$
$A'B$	$m_4$	$m_5$	$m_7$	$m_6$
$A'B$	$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
$A'B'$	$m_8$	$m_9$	$m_{11}$	$m_{10}$

(b)

AB \ CD	00	01	11	10
00	$A'B'C'D'$	$A'B'C'D$	$A'B'CD$	$A'B'CD'$
01	$A'BC'D'$	$A'BC'D$	$A'BCD$	$A'BCD'$
11	$ABC'D'$	$ABC'D$	$ABCD$	$ABCD'$
10	$AB'C'D'$	$AB'C'D$	$AB'CD$	$AB'CD'$

(c)

Tabla 17. Mapa de cuatro variables

Por ejemplo, la celda  $m_9$  corresponde al mintérmino 9, ubicado en la fila 10 y la columna 01. La unión de estos dos números da el número 1001, cuyo equivalente es el término  $A'B'C'D$  -ó el decimal 9.

La minimización por medio de un mapa de 4 variables se puede efectuar con las celdas adyacentes entre sí y las celdas de los bordes que se pueden concatenar para reducir la expresión. Por ejemplo,  $m_{13}$  y  $m_{15}$  son celdas adyacentes así como  $m_0$ ,  $m_8$ ,  $m_2$  y  $m_{10}$ .

El mapa se construye colocando un 1 en las celdas correspondientes a los mintérminos presentes en la función de salida. Por ejemplo, para el término  $F(1,1,0,0) = A \cdot B \cdot C' \cdot D' = 1$  se situaría un 1 en la celda 1100. Para los mintérminos no presentes en la función se pone un 0. Por ejemplo el término  $F(1,1,1,1) = A \cdot B \cdot C \cdot D = 0$ , será una celda con valor 0 en la celda 1111.

Igual que en el mapa de 3 variables, se procede con la agrupación de 1s, la determinación del término producto correspondiente a cada grupo y la suma de los términos producto obtenido.

Las reglas para reducir términos en un mapa de Karnaugh de 4 variables son las siguientes:

1. Una celda representa un mintérmino, dando como resultado un término de cuatro literales.
2. Dos celdas agrupadas pueden representar la asociación de dos mintérminos, dando como resultado un término de tres literales.
3. Cuatro celdas agrupadas pueden representar la asociación de cuatro mintérminos, dando como resultado un término de dos literales.
4. Ocho celdas agrupadas pueden representar la asociación de ocho mintérminos, dando como resultado un término de un literal.
5. Dieciséis celdas agrupadas pueden representar un valor de función igual a 1.

*Ejemplo* Simplíquese la función de Boole  $F_2 = \sum (m_1, m_3, m_8, m_{10}, m_{12}, m_{14})$

CD AB	00	01	11	10
	00	01	11	10
00	0	1	1	0
01	0	0	0	0
11	1	0	0	1
10	1	0	0	1

**Figura 33.** Mapa de Karnaugh de la función  $F_2$ .

El primer grupo se forma con los mintérminos  $m_1$  y  $m_3$  y el segundo grupo se forma con los mintérminos  $m_8, m_{10}$  y  $m_{12}, m_{14}$ .

Del primer grupo resulta el término  $A' \cdot B' \cdot D$  ya que en la columna 1 no se presentan cambios para las variables A y B y se presenta transición en la variable C en las columnas 2 y 3. El segundo grupo da como resultado el término  $A \cdot D'$ . La razón radica en la simplificación de la variable B en la tercera y cuarta fila y en la variable C en la primera y cuarta columna.

Sumando los mintérminos obtenidos se obtiene la ecuación simplificada:  $F_2 = A' \cdot B' \cdot D + A \cdot D'$

### Mapas de Karnaugh empleando Producto de Sumas (PDS)

La simplificación de expresiones lógicas mediante el mapa de Karnaugh también es posible mediante el método de producto de sumas. En este método, cada celda representa un maxtérmino.

La construcción del mapa es similar a la suma de productos. La diferencia radica en que cada celda representa un maxtérmino. Por ejemplo, la celda  $m_2$  corresponde al maxtérmino 2, ubicado en la fila 0 y la columna 10. La unión de estos dos números da el número 010, cuyo equivalente es el término  $A+B'+C$ . La figura 34. Muestra el mapa de Karnaugh para 3 variables.

A \ BC	00	01	11	10
	0	1	1	0
0	$A+B+C$	$A+B+C'$	$A+B'+C'$	$A+B'+C$
1	$A'+B+C$	$A'+B+C'$	$A'+B'+C'$	$A'+B'+C$

Figura 34. Mapa de tres variables.

La representación de la función lógica se hace simplemente copiando los ceros de la tabla de verdad en las celdas del mapa. Este método es más apropiado cuando en la columna de resultados de la tabla de verdad predominan los ceros.

*Ejemplo* Utilizar el mapa de Karnaugh para minimizar el producto de sumas,

$$F_3 = (A+B+C) \cdot (A'+B+C) \cdot (A+B'+C) \cdot (A'+B'+C)$$

Los maxtérminos se trasladan a cada una de las celdas del mapa de Karnaugh y las celdas se agrupan tal como en la figura 35.

A \ BC	00	01	11	10
	0	1	1	0
0	0	1	1	0
1	0	1	1	0

Figura 35. Mapa de Karnaugh de la función  $F_3$

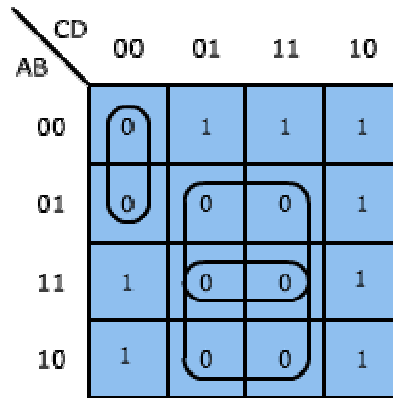
El término suma para cada grupo se muestra en la figura y la suma de productos resultante es:

$$F_3 = C$$

*Ejemplo* Utilizar el mapa de Karnaugh para minimizar el producto de sumas,

$$F_4 = (A+B+C+D) \cdot (A+B'+C) \cdot (A+B'+C'+D') \cdot (A'+B'+C+D') \cdot (A'+B+C'+D') \cdot (A'+B+C+D') \cdot (A'+B+C+D') \cdot (A'+B'+C+D')$$

El segundo término tiene que ampliarse a  $(A+B'+C+D) \cdot (A+B'+C+D')$ . La función completa se pasa al mapa de karnaugh mostrado en la figura 36.



**Figura 36.** Mapa de Karnaugh de la función  $F_4$

El término suma para cada grupo se muestra en la figura 2.4.5. y el producto de sumas resultante es:

$$F_4 = (A+C+D) \cdot (B'+D') \cdot (A'+D')$$

### Condiciones de No Importa

Hasta el momento se ha asumido que la función es igual a 0 en los casos donde la función no es igual a 1. En algunas aplicaciones esta suposición no es siempre verdadera ya que existen combinaciones de entrada que no presentan. En un mapa de Karnaugh estas combinaciones de entrada sirven de herramienta para simplificar la función y su representación se hace por medio de una X en la celda del mapa. Según la agrupación que convenga se asume un valor de 1 ó 0 para la X con el fin de obtener la expresión más simple.

### Ejemplo

Simplificar la función de Boole  $F_5 = \Sigma (m_0, m_4, m_7, m_9)$  con condiciones de importa,

$$NI = \Sigma (m_1, m_5, m_{11}, m_{14}).$$

Los minterminos se marcan con un 1, las condiciones de no importa con una X y las celdas restantes con 0.

El mapa de Karnaugh de la función  $F_5$  se muestra en la figura 37.

CD \ AB	00	01	11	10
00	1	X	0	0
01	1	0	1	0
11	0	0	0	X
10	0	1	X	0

Figura 37. Mapa de Karnaugh de la función  $F_5$ 

En suma de productos obtenemos,  $F_5 = A' \cdot C' \cdot D' + A' \cdot B' \cdot C' + A' \cdot B \cdot C \cdot D + A \cdot B' \cdot D$

#### d. Algoritmo de Quine – McCluskey

El empleo del mapa de Karnaugh es conveniente cuando la función a minimizar no contiene más de cinco o seis variables. En estos casos, empleamos un procedimiento sistemático, llamado el algoritmo de Quine–McCluskey, el cual produce una expresión normalizada y simplificada. El algoritmo debe obedecer a un conjunto de pasos que se verán a través de un ejemplo.

*Ejemplo* Simplificar la función de Boole usando el algoritmo de Quine-McCluskey.

$$F_1 = \sum (m_1, m_2, m_3, m_6, m_7, m_8, m_9, m_{10}, m_{15})$$

$$F_1 = A' \cdot B' \cdot C' \cdot D + A' \cdot B' \cdot C \cdot D' + A' \cdot B' \cdot C \cdot D + A' \cdot B \cdot C \cdot D' + A' \cdot B \cdot C \cdot D + A \cdot B' \cdot C' \cdot D' + A \cdot B' \cdot C' \cdot D + A \cdot B' \cdot C \cdot D' + A \cdot B \cdot C \cdot D.$$

1. Enumerar en una tabla todos los minterminos en forma binaria, organizados según el número de unos que contenga. La aplicación de este paso se muestra en la tabla 18.

Mintérminos	A	B	C	D	Grupo
1	0	0	0	1	Grupo 1
2	0	0	1	0	
8	1	0	0	0	
3	0	0	1	1	Grupo 2
6	0	1	1	0	
9	1	0	0	1	
10	1	0	1	0	
7	0	1	1	1	Grupo 3
15	1	1	1	1	Grupo 4

Tabla 18. Mintérminos agrupados según la cantidad de unos

- Entre los grupos adyacentes buscar los mintérminos que sólo difieren en un bit en la misma posición, para hallar los primeros implicantes primos.

La metodología consiste en comparar el primer mintérmino con el resto de los términos del segundo grupo. Así, los términos del segundo grupo se comparan con los mintérminos del grupo siguiente. De la forma anterior, se procede con los demás mintérminos de los demás grupos. Los mintérminos utilizados se les pone una marca (✓) con el fin de ir diferenciando los términos utilizados y la variable apareada en el proceso se reemplaza con un guión para denotar la eliminación de la variable. Los términos no marcados en la tabla son los primeros implicantes primos ( $PI_x$ ). Los mintérminos utilizados se les pone una marca (✓) con el fin de ir diferenciando los términos utilizados y la variable apareada en el proceso anterior se reemplaza con un guión para denotar la eliminación de la variable.

Mintérmino	A	B	C	D	Mintérmino	A	B	C	D	$PI_x$	Mintérmino	A	B	C	D	$PI_x$
1 ✓	0	0	0	1	1-3	0	0	-	1	$PI_2$	2-6 - 3-7	0	-	1	-	$PI_1$
2 ✓	0	0	1	0	1-9	-	0	0	1	$PI_3$	2-3 - 6-7	0	-	1	-	
8 ✓	1	0	0	0	2-3 ✓	0	0	1	-							
3 ✓	0	0	1	1	2-6 ✓	0	-	1	0							
6 ✓	0	1	1	0	2-10	-	0	1	0	$PI_4$						
9 ✓	1	0	0	1	8-9	1	0	0	-	$PI_5$						
10 ✓	1	0	1	0	8-10	1	0	-	0	$PI_6$						
7 ✓	0	1	1	1	3-7 ✓	0	-	1	1							
15 ✓	1	1	1	1	6-7 ✓	0	1	1	-							
					7-15	-	1	1	1	$PI_7$						

**Tabla 19.** Implicantes primos de la función  $F_1$

- Construir una tabla que enumere los implicantes primos y los mintérminos contenidos por cada implicante primo. La letra X en la tabla 20 indica el mintérmino contenido en cada implicado por fila. Por ejemplo, en la tabla se observa en el primer renglón los mintérminos 2, 3, 6 y 7 para el primer implicante primo. El resto de la tabla se construye de forma similar.

Implicante Primo	1	2	3	6	7	8	9	10	15
* $PI_1$		X	X	X	X				
$PI_2$	X		X						
$PI_3$	X						X		
$PI_4$		X						X	
$PI_5$						X	X		
$PI_6$						X		X	
* $PI_7$					X				X

**Tabla 20.** Selección de implicantes primos esenciales



En la tabla se seleccionan las columnas de los minterminos que contengan solamente una cruz. En este ejemplo, hay dos minterminos cuyas columnas tienen una sola cruz: 6 y 15. Es decir, la selección del primer implicado  $PI_1 (A' \cdot C)$  garantiza que el término mínimo 6 está incluido en la función. De la misma forma, el término mínimo 7 está cubierto por el primer implicado  $PI_7 (A' \cdot B \cdot C \cdot D)$ . Los primeros implicados que cubren los minterminos con una sola cruz, se llaman primeros implicados esenciales (en la tabla se encuentran marcados con un asterisco) y son indispensables en la construcción de la función.

4. Seleccionar en cada columna los minterminos que estén cubiertos por los primeros implicados esenciales. Por ejemplo, el primer implicado esencial  $*PI_1 (A' \cdot C)$  cubre los minterminos 2, 3, 6 y 7. De la misma forma, el primer implicado esencial  $*PI_7 (A' \cdot B \cdot C \cdot D)$  cubre los minterminos 7 y 15. Hasta el momento la selección de primeros implicados cubre los minterminos 2, 3, 6, 7 y 15 excepto 1, 8, 9 y 10. Estos términos mínimos deben ser seleccionados por medio de otros primeros implicados esenciales. En la tabla 2.5., la selección de los primeros implicados  $PI_3$  y  $PI_6$  garantiza el cubrimiento de los términos mínimos 1, 8, 9 y 10. En la tabla 21 se muestra el proceso de selección.

Implicante Primo	1	8	9	10
$PI_2$	X			
$*PI_3$	X		X	
$PI_4$				X
$PI_5$		X	X	
$*PI_6$		X		X

**Tabla 21.** Selección de primeros implicados esenciales

La función simplificada se obtiene de la suma de los primeros implicados hallados:

$$F = PI_1 + PI_3 + PI_6 + PI_7$$

$$F = (0-1-) + (-001) + (10-0) + (-111)$$

$$F = A' \cdot C + B' \cdot C' \cdot D + A \cdot B' \cdot D' + B \cdot C \cdot D$$

### 3. Circuitos Lógicos Combinacionales

Los circuitos lógicos se dividen en combinacionales y secuenciales. Los circuitos combinacionales consisten en variables de entrada, compuertas lógicas y variables de salida que cumplen funciones intermedias de mediana escala de integración. El nivel de complejidad de los sistemas combinacionales puede llegar al caso de millones de entradas, dispositivos, interconexiones y salidas. La comprensión de estos circuitos se hace por medio de la división en subsistemas o estructuras más simples.

Hay esencialmente tres tipos de funciones en el diseño de circuitos lógicos combinacionales:

- a. **Funciones aritmético lógicas (ALU):** Encargados de realizar operaciones locales entre dos datos de  $n$  bits. (Sumadores, restadores, multiplicadores y operaciones lógicas bit a bit).
- b. **Funciones de ruta de datos:** Guían el tráfico de datos e instrucciones entre las distintas partes de un sistema de cálculo (de memoria a unidad aritmética, etc,...). Su clave es el carácter controlado del movimiento a través de compuertas que se abren o cierran de forma sincrónica, en general, de acuerdo con pulsos de un reloj. (multiplexores, demultiplexores).
- c. **Circuitos cambiadores de código:** Para cada tipo de proceso existe una representación digital de la información que es más adecuada que otras. Su ejemplo más general es el de las memorias de solo lectura (ROM), que son en realidad circuitos que sintetizan funciones múltiples de forma que cada bit de la palabra de salida puede ser una función lógica cualquiera de todos los bits de entrada.

#### 3.1. Circuitos Aritméticos

El diseño de sistemas digitales involucra el manejo de operaciones aritméticas. A continuación se implementarán los circuitos de suma y resta de números binarios.

La suma o adición binaria es análoga a la de los números decimales. La diferencia radica en que en los números binarios se produce un acarreo (carry) cuando la suma excede de uno mientras en decimal se produce un acarreo cuando la suma excede de nueve(9). Del gráfico de la figura 1 podemos sacar las siguientes conclusiones:

1. Los números o sumandos se suman en paralelo o en columnas, colocando un número encima del otro. Todos los números bajo la misma columna tienen el mismo valor posicional.
2. El orden de ubicación de los números no importa (propiedad conmutativa).

En la figura 38 se indican las reglas que rigen la suma binaria y en la figura 40 se muestra un circuito lógico llamado semisumador, que suma 2 bits (A y B) que genera un bit de suma y un bit de acarreo cuando este se produce.

Regla 1  $0 + 0 = 0$   
 Regla 2  $0 + 1 = 1$   
 Regla 3  $1 + 0 = 1$   
 Regla 4  $1 + 1 = 0$  y arrastre  $1 = 10$

**Figura 38.** Reglas para la suma binaria

La resta o sustracción de números binarios es similar a los números decimales. La diferencia radica en que, en binario, cuando el minuendo es menor que el sustraendo, se produce un préstamo o borrow de 2, mientras que en decimal se produce un préstamo de 10.

Al igual que en la suma, el proceso de resta binaria, se inicia en la columna correspondiente a la de los dígitos menos significativos. En la figura 39 se indican las reglas que rigen la resta binaria.

	0	0			Préstamo
1	<del>1</del> → (1)0	<del>1</del> → (1)0			Minuendo
-0	0	1	0	1	Sustraendo
<hr/>					
1	0	1	0	1	Diferencia

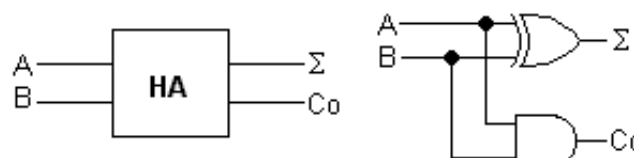
**Figura 39.** Reglas para la resta binaria

**Semisumador:** La operación de un Semisumador conforme a las reglas de la suma binaria mostradas en la figura 38 se puede sintetizar mediante las siguientes 2 operaciones booleanas:

$$\Sigma = A(\text{xor})B \text{ (suma)} \quad Co = A \cdot B \text{ (acarreo)}$$

Para realizar una suma binaria donde se tenga presente un carry de entrada se debe implementar un circuito que tenga presente esta nueva variante; como es el caso del sumador completo (Figura 42).

El bit de suma  $\Sigma$  es 1, sólo si las variables  $A$  y  $B$  son distintas. El bit de acarreo  $Co$  es 0 a no ser que ambas entradas sean 1. Por consiguiente, la salida  $S$  puede expresarse en términos de la operación  $EXOR$ :  $\Sigma = A' \cdot B + A \cdot B' = A \oplus B$



**Figura 40.** Símbolo y Circuito Lógico Semisumador

**Sumador Completo:** El sumador completo acepta dos bits y un acarreo de entrada y genera una suma de salida junto con el acarreo de salida. El sumador completo tiene 3 entradas que se suman y son: A, B, y  $C_{in}$  (entrada de arrastre), y las salidas habituales  $\Sigma$  y  $C_{out}$  (suma y salida de arrastre)

La tabla 22 muestra la tabla de verdad del sumador completo. Las entradas A, B y  $C_{in}$  denotan al primer sumando, el segundo sumando y el acarreo de entrada. Las salidas S y  $C_{out}$  representan a la suma y el acarreo de salida.

A	B	$C_{in}$	$C_{out}$	$\Sigma$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

**Tabla 22.** Tabla de Verdad del Sumador Completo

$$\Sigma = A \cdot B' \cdot C_{in}' + A' \cdot B \cdot C_{in}' + A \cdot B \cdot C_{in} + A' \cdot B' \cdot C_{in}$$

$$\Sigma = C_{in}' \cdot (A \cdot B' + A' \cdot B) + C_{in} \cdot (A \cdot B + A' \cdot B')$$

$$\Sigma = C_{in}' \cdot (A \cdot B' + A' \cdot B) + C_{in} \cdot (A' \cdot A + A' \cdot B' + A \cdot B + B \cdot B')$$

$$\Sigma = C_{in}' \cdot (A \cdot B' + A' \cdot B) + C_{in} \cdot ((A' + B) \cdot (A + B'))$$

$$\Sigma = C_{in}' \cdot (A \cdot B' + A' \cdot B) + C_{in} \cdot ((A \cdot B')' \cdot (A' \cdot B)')$$

$$\Sigma = C_{in}' \cdot (A \cdot B' + A' \cdot B) + C_{in} \cdot (A \cdot B' + A' \cdot B)'$$

$$\Sigma = (A \oplus B) \oplus C_{in}$$

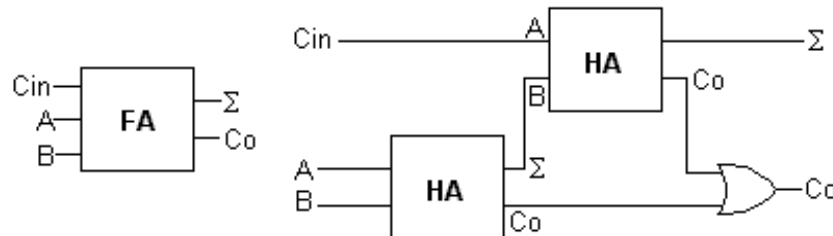
El mapa de karnaugh de la salida  $C_{out}$  se muestra en la figura 41.

A \ BC	BC			
	00	01	11	10
0	0	0	1	0
1	0	1	1	1

**Figura 41.** Mapa para la salida  $C_{out}$  de un Sumador Completo.

La salida  $C_{out}$  está dada por:

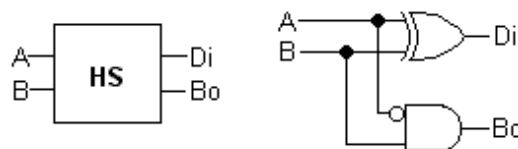
$$C_{out} = A \cdot B + A \cdot C_{in} + B \cdot C_{in}$$

**Figura 42.** Símbolo y Circuito Lógico Sumador Completo

**Semirrestador:** La operación de un Semirrestador como el mostrado en la figura 43 se puede resumir mediante las ecuaciones booleanas:

$$Di = A \cdot B(\text{neg}) + A(\text{neg}) \cdot B = A(\text{xor})B \text{ (diferencia)} \quad Bi = A(\text{neg}) \cdot B \text{ (borrow)}$$

El circuito tiene dos entradas binarias y dos salidas. La figura 43 muestra el símbolo lógico y el circuito las entradas son  $A$  (minuendo) y  $B$  (sustraendo) y la salida  $Di$  corresponde a la diferencia y  $Bo$  al préstamo de salida.

**Figura 43.** Símbolo y Circuito Lógico Semirrestador

Si  $A \geq B$ , existen tres posibilidades  $0-0=0$ ,  $1-0=1$  y  $1-1=0$ . El resultado es el bit de diferencia  $Di$ . Si  $A < B$  se tiene  $0-1$  y es necesario prestar un 1 de la siguiente posición significativa de la izquierda. El préstamo agrega 2 al bit del minuendo de manera similar cuando en el sistema decimal se agrega 10 al dígito del minuendo.

La tabla de verdad 23. Está dada por las reglas de la resta binaria.

A	B	Bo	Di
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

**Tabla 23.** Tabla de verdad del Restador medio.

La salida  $Di$  coincide con la operación  $EXOR$  y se puede expresar de la siguiente forma:

$$Di = A' \cdot B + A \cdot B'$$

La salida  $Bo$  está dada por la suma de productos de los términos presentes en el renglón 2 de la tabla de verdad:

$$Bo = A' \cdot B$$

### Restador Completo

El Restador completo realiza la resta entre dos bits, considerando que se ha prestado un 1 de un estado menos significativo. En la tabla 24 las entradas  $A$ ,  $B$  y  $C$  denotan el minuendo, el sustraendo y el bit prestado. Las salidas  $Di$  y  $Bo$  representan a la diferencia y el préstamo.

A	B	C	Bo	Di
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

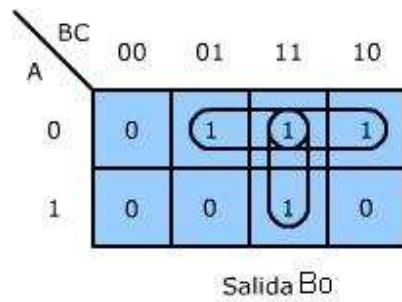
**Tabla 24.** Tabla de verdad del Restador Completo.

En las combinaciones del mapa donde  $C=0$ , se tienen las mismas condiciones para el semisumador.

La función de la salida  $Di$  de un restador es la misma que la salida de un sumador completo:

$$D = A' \cdot B' \cdot C + A' \cdot B \cdot C' + A \cdot B' \cdot C' + A \cdot B \cdot C = (A \oplus B) \oplus C_{in}$$

El mapa de karnaugh de la salida  $Bo$  se muestra en la figura 44.

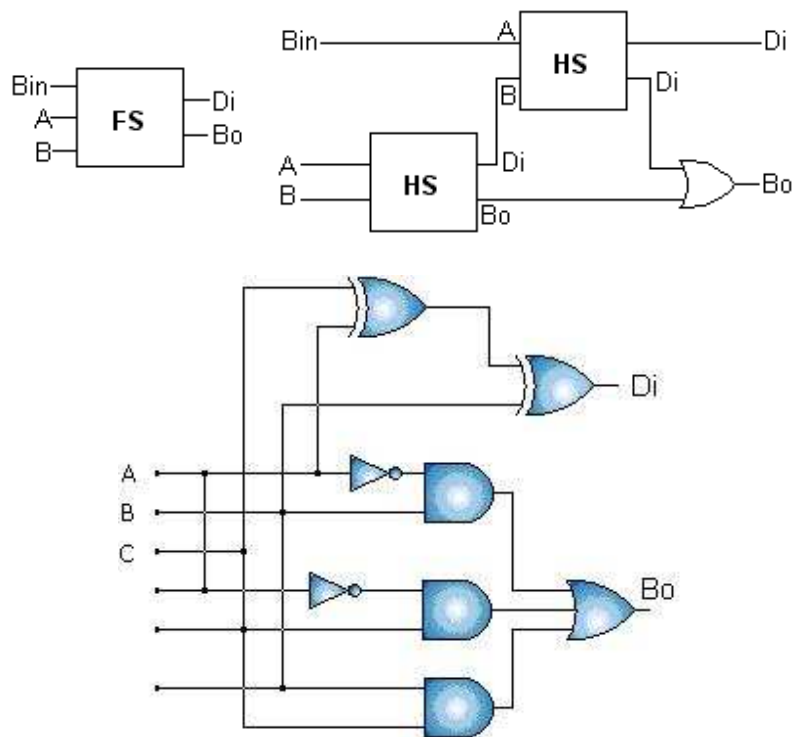


**Figura 44.** Mapa para la salida  $B_0$  de un restador completo

La salida  $B_0$  está dada por:

$$P = A' \cdot B + A' \cdot C + B \cdot C$$

El símbolo y circuito lógico se muestra en la figura 45.



**Figura 45.** Símbolo y Circuito Lógico Restador Completo

### Sumador y Restador de Cuatro Bits

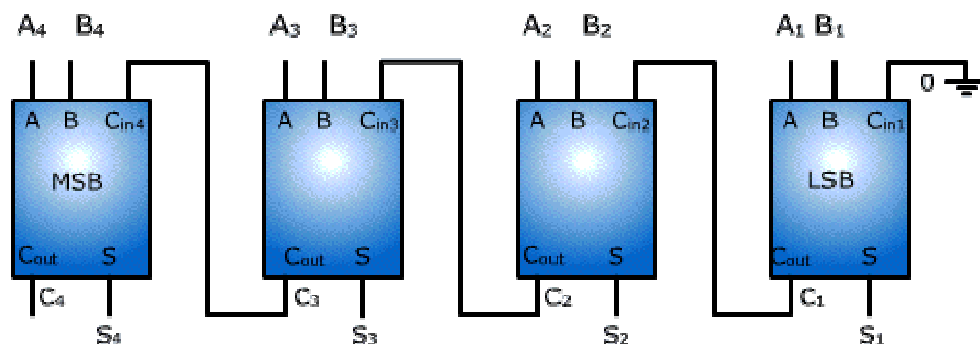
Las operaciones aritméticas se pueden implementar mediante circuitos lógicos. El nivel de sencillez obtenido en los circuitos está dado por la técnica de diseño utilizada. La implementación de una unidad aritmética que realice las operaciones de suma y resta en un sólo circuito, es más simple comparándola con una de dos circuitos para las mismas funciones.

La suma de dos números binarios de cuatro bits se realiza de derecha a izquierda, teniendo en cuenta las correspondientes posiciones significativas y el bit de arrastre (acarreo  $C_{inx}$ ). El bit de arrastre generado en cada posición se utiliza en la siguiente posición significativa. La figura 46 muestra la suma de dos números de cuatro bits.

	$C_{in4}$	$C_{in3}$	$C_{in2}$	$C_{in1}$	Acarreo de Entrada $C_{in}$
	$A_4$	$A_3$	$A_2$	$A_1$	Sumando A
+	$B_4$	$B_3$	$B_2$	$B_1$	Sumando B
	$S_4$	$S_3$	$S_2$	$S_1$	

**Figura 46.** Suma binaria de cuatro bits

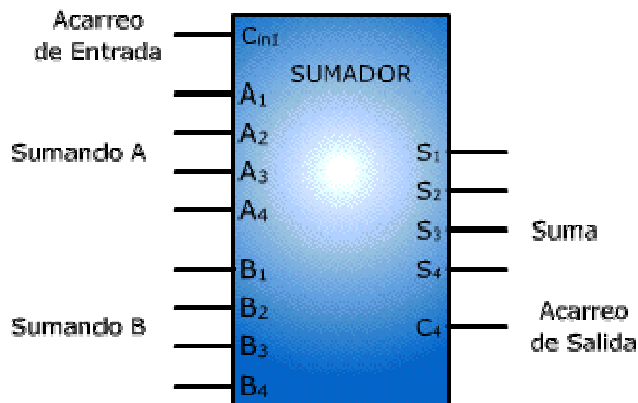
En un sumador completo, la suma de un par de bits genera un bit de acarreo. Un sumador de 2 números de  $n$  bits se puede implementar de la forma descrita a continuación. Los bits de la posición menos significativa se suman con un acarreo inicial de 0, generando el bit de suma y el de acarreo. El bit de acarreo generado es usado por el par de dígitos en la siguiente posición significativa. La suma se propaga de derecha a izquierda según los acarreos generados en cada sumador y los sumandos presentes. Por consiguiente, la suma de dos 2 números binarios de  $n$  bits se puede implementar mediante la utilización de  $n$  sumadores completos. Así, para números binarios de dos bits se necesitan dos sumadores completos; para números de cuatro bits cuatro sumadores. En la figura 47 se muestra un sumador de cuatro bits.



**Figura 47.** Símbolo lógico del sumador en paralelo de cuatro bits

El símbolo lógico del sumador de cuatro bits se muestra en la figura 48.

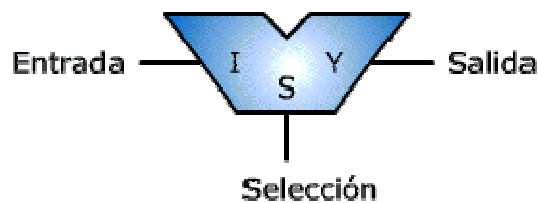




**Figura 48.** Circuito lógico del sumador en paralelo de cuatro bits

Un sumador se puede modificar en forma de sustractor invirtiendo cada bit del sustraendo y sumando 1 al establecer un acarreo de entrada  $C_{in1}$ .

Observe el complementador de la figura 49. Si la entrada de control es igual a  $S=0$ , la entrada de datos  $I$  pasa sin ningún cambio a la salida. Si  $S=1$ , la entrada de datos se complementa.



**Figura 49.** Diagrama de bloque de un complementador

El funcionamiento de este elemento se describe en la tabla de verdad 25.

Entradas		Salida	Descripción
S	I	Y	
0	0	0	Pasa a Y
0	1	1	Pasa a Y
1	0	1	Complemento a Y
1	1	0	Complemento a Y

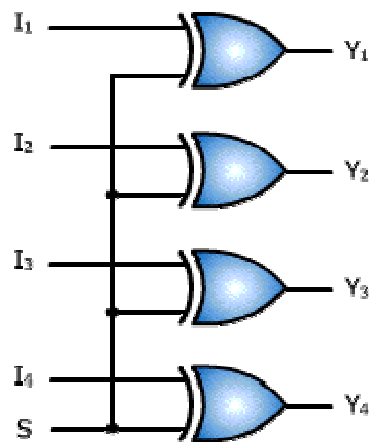
**Tabla 25.**Tabla de verdad de un complementador

De la tabla de verdad se observa que  $Y = S \oplus I$ . La figura 50 muestra la función *EXOR* como complementador.



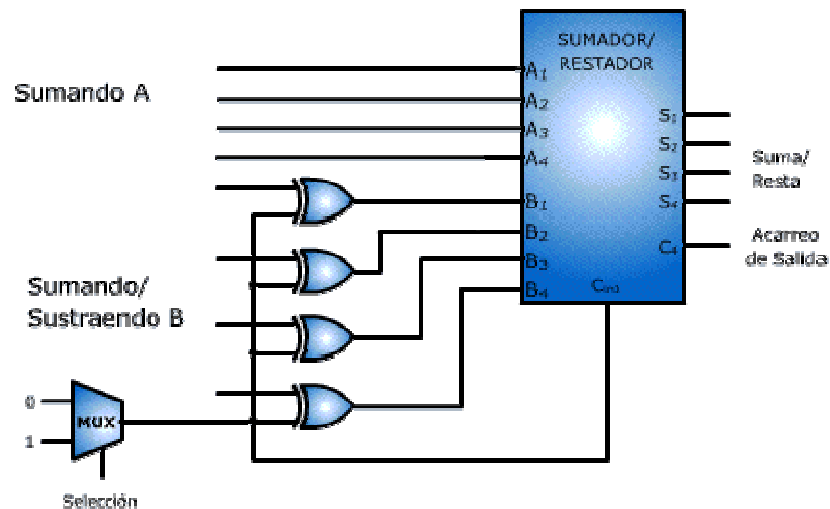
**Figura 50.** Función EXOR como complementador

Una sola entrada de control  $S$  con  $n$  líneas de entrada de datos  $I$ , sirve para complementar o no complementar la entrada, según la operación de resta o suma binaria. La figura 51 ilustra un complementador de 4 bits.



**Figura 51.** Complementador de 4 bits

El circuito completo de un sumador/restador de 4 bits se representa en la figura 52.



**Figura 52.** Sumador/restador de 4 bits

## Sumador en BCD

La suma en código *BCD* utiliza las mismas reglas de la suma binaria vistas en la figura 38. Si una suma de dos números es menor o igual que 9, el número *BCD* resultante es válido. Si la suma es mayor que 9, o si se genera un acarreo el resultado no es válido. En este caso, se suma el número binario *0110* para pasar de nuevo al código *BCD*. Si se genera acarreo al sumar *0110*, éste se suma al siguiente grupo de 4 bits. En los siguientes ejemplos se verán los casos que se pueden presentar.

Ejemplo Sumar los números  $01000101 (45)_{10}$  y  $00010010 (12)_{10}$ .

La suma de la figura 53 no genera acarreos.

$$\begin{array}{r}
 0100 \quad 0101 \quad 45 \\
 + 0001 \quad 0010 \quad +12 \\
 \hline
 0101 \quad 0111 \quad 57
 \end{array}$$

**Figura 53.** Suma BCD sin acarreo.

Ejemplo Sumar los números  $00111001 (39)_{10}$  y  $01010110 (56)_{10}$ .

La suma de los cuatro bits menos significativos de la figura 54 genera acarreo.

$$\begin{array}{r}
 \overset{1}{0} \overset{1}{0} \overset{1}{1} 1 \quad 1001 \quad 39 \\
 + 0101 \quad 0110 \quad +56 \\
 \hline
 1000 \quad 1111 \quad 95 \\
 \hline
 1001 \quad 0101
 \end{array}$$

**Figura 54.** Suma BCD con acarreo en el dígito BCD menos significativo

Ejemplo Sumar los números  $01111001 (79)_{10}$  y  $00110101 (35)_{10}$ .

La suma de dígito BCD menos significativo de la figura 55 genera acarreo, al igual que el segundo dígito BCD.

$$\begin{array}{r}
 \overset{1}{0} \overset{1}{0} \overset{1}{1} 1 \quad 1001 \quad 79 \\
 + 0011 \quad 0101 \quad +35 \\
 \hline
 1010 \quad 1100 \quad 114 \\
 \hline
 0110 \quad 0110 \\
 \hline
 0001 \quad 0001 \quad 0101 \\
 \hline
 \underbrace{0001}_1 \quad \underbrace{0001}_1 \quad \underbrace{0101}_4
 \end{array}$$

**Figura 55.** Suma BCD con acarreo en dos dígitos

Un sumador *BCD* es un circuito que suma dos dígitos en *BCD*. En una suma *BCD*, la suma  $9+9+1=19$  es el valor máximo resultante, siendo el 1 en la suma el acarreo de entrada. Los dígitos *BCD* con un acarreo de entrada, se agregan en un sumador binario de cuatro bits para producir la suma binaria. Los números decimales se listan en la tabla 3.11.1.  $C_1$  es el acarreo de la suma de los números *A* y *B* de entrada (ver figura 56) y los dígitos  $S_1$  a  $S_4$  son el resultado de la suma binaria, donde cada dígito tiene los pesos 8, 4, 2, 1 del código *BCD*. Cuando la suma binaria es menor o igual a 1001, no se agrega nada a la suma. Cuando el número binario es mayor que 1001 se obtiene una representación en código *BCD* no válida. La suma del número binario 0110 a la suma binaria convierte la representación a un código *BCD* válido. En la figura la suma del número 0110 se realiza por medio de un segundo sumador inferior. Este código *BCD* válido se observa en la tabla 3.11.1 en la columna de suma *BCD*. Las salidas  $S_5$  a  $S_8$  representan la suma *BCD*.  $C_2$  es el acarreo de salida de la suma *BCD*.

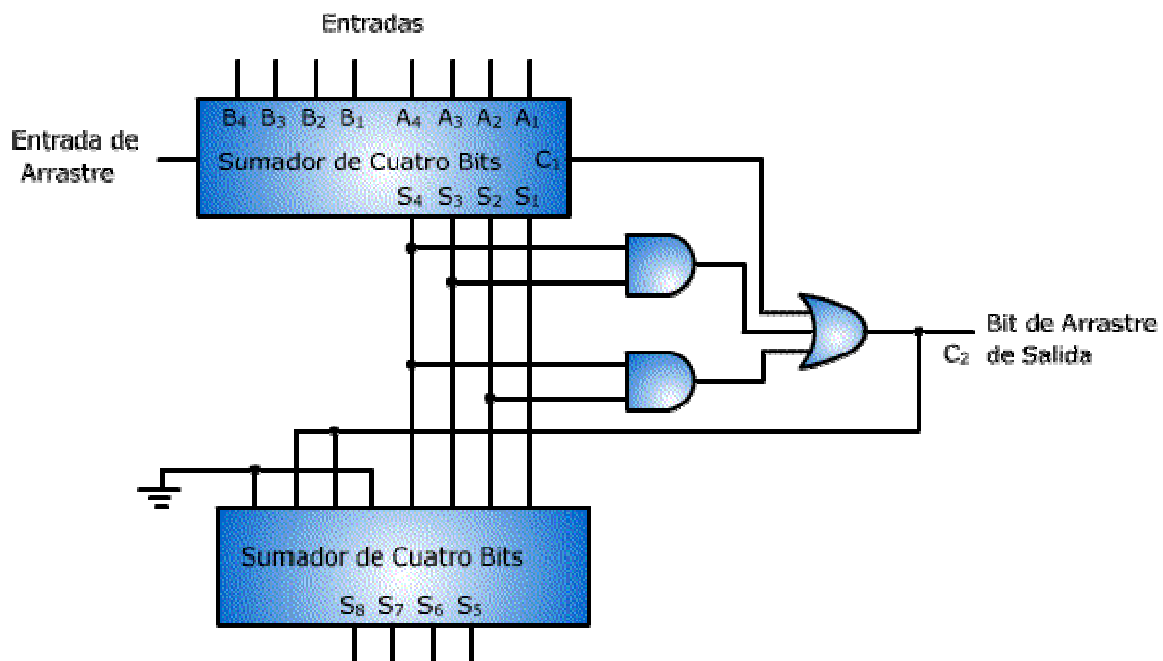
Decimal	Suma Binaria					Suma BCD				
	$C_1$	$S_4$	$S_3$	$S_2$	$S_1$	$C_2$	$S_8$	$S_7$	$S_6$	$S_5$
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

**Tabla 26.** Tabla de verdad del Sumador BCD.

El circuito necesario para detectar la condición de acarreo o suma binaria mayor a 1001 se obtiene de la tabla de verdad. Cuando  $C_1$  es 1 se necesita sumar 0110 o una corrección. Lo mismo entre las combinaciones 1010 y 1111, se tiene una corrección cuando  $S_2=S_4=1$  ó  $S_3=S_4=1$ . La expresión lógica de la corrección es:

$$C_2 = C_1 + S_3 \cdot S_4 + S_4 \cdot S_2$$

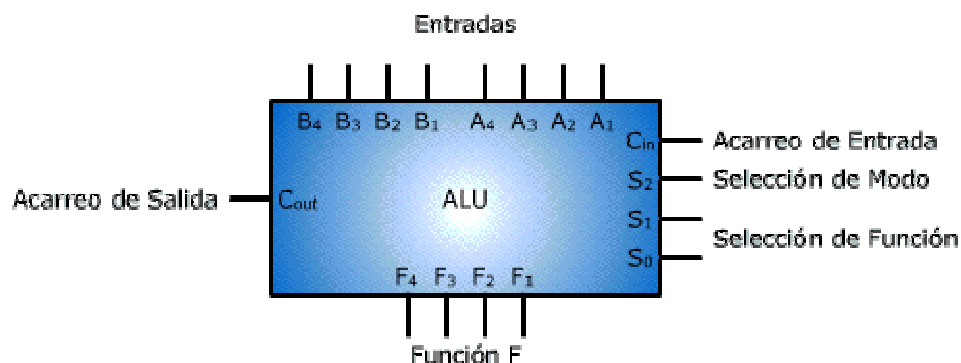
El circuito lógico necesario para implementar el sumador *BCD* se muestra en la figura 56



**Figura 56.** Diagrama de bloques de un sumador BCD

### Unidad Aritmética y Lógica (ALU)

Una unidad aritmética lógica puede realizar un conjunto de operaciones aritméticas básicas y un conjunto de operaciones lógicas, a través de líneas de selección. En inglés *ALU* significa *Arithmetic Logic Unit* (Unidad Aritmética Lógica). La figura 57. Muestra el diagrama de bloques de una *ALU*.



**Figura 57.** Diagrama de bloques de una ALU

Las cuatro entradas de  $A$  se combinan con las de  $B$  generando una operación de salida de cuatro bits en  $F$ . La entrada de selección de modo  $S_2$  distingue entre las operaciones aritméticas y lógicas. Las entradas de selección  $S_0$  y  $S_1$  determinan la operación aritmética o lógica. Con las entradas  $S_0$  y  $S_1$  se pueden elegir cuatro operaciones aritméticas (con  $S_2$  en un estado) y cuatro lógicas (con  $S_2$  en otro estado). Los acarrees de entrada y salida tienen sentido únicamente en las operaciones aritméticas. El diseño de una  $ALU$  implica el diseño de la sección aritmética, la sección lógica y la modificación de la sección aritmética para realizar las operaciones aritméticas y lógicas.

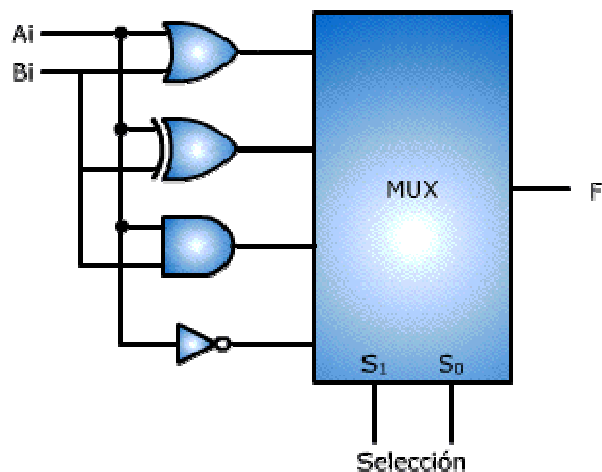
### Sección Lógica

Los datos de entrada en una operación lógica son manipulados en forma separada y los bits son tratados como variables binarias. En la tabla 27 se listan cuatro operaciones lógicas  $OR$ ,  $EXOR$ ,  $AND$  y  $NOT$ . En el circuito, las dos líneas de selección ( $S_1$ ,  $S_0$ ) permiten seleccionar una de las compuertas de entrada, correspondientes a la función  $F_i$ .

$S_1$	$S_0$	Salida	Función $F_i$
0	0	$F=A_i+B_i$	OR
0	1	$F=A_i \oplus B_i$	XOR
1	0	$F=A_i \cdot B_i$	AND
1	1	$F=A'_i$	NOT

**Tabla 27.** Tabla de Función Lógica.

El circuito lógico de la figura 58 es una etapa de un circuito lógico de  $n$  bits.



**Figura 58.** Diagrama lógico de un circuito lógico de una ALU

### Sección Aritmética

El componente básico de la sección aritmética es un sumador en paralelo (ver figura 47). Las operaciones aritméticas configuradas en el circuito aritmético se presentan en la tabla

28. En una *ALU*, la suma aritmética se puede implementar con un número binario en *A*, otro número en la entrada *B* y el acarreo de entrada  $C_{in}$  en un valor lógico 0. El resto de las funciones se enuncian en la columna descripción.

Selección de Función			Salida N	Función	Descripción
$S_I$	$S_0$	$C_{in}$	$N$	$F$	
0	0	0	0	A	Transferir A
0	0	1	0	A+1	Incrementar A
0	1	0	B	A+B	Suma ó agregar B a A
0	1	1	B	A+B+1	Suma con acarreo ó agregar B a A más 1
1	0	0	B'	A+B'	Agregar el complemento de 1 de B a A
1	0	1	B'	A+B'+1	Agregar el complemento de 2 de B a A
1	1	0	Todos unos	A-1	Decrementar A
1	1	1	Todos unos	A	Trasferir A

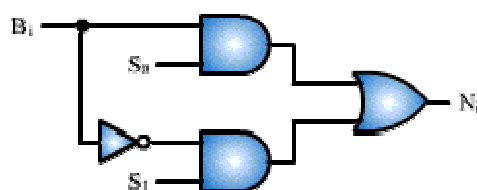
**Tabla 28.** Tabla de la Función F en un Circuito Aritmético

La implementación de las funciones anteriores por medio de un circuito lógico sencillo se describe a continuación. El circuito se diseña bajo el precepto de intervenir cada entrada  $B_i$  para obtener las siguientes funciones:

$S_I$	$S_0$	$N_i$
0	0	0
0	1	$B_i$
1	0	$B_i'$
1	1	1

**Tabla 29.** Tabla del circuito para la entrada  $B_i$

La figura 59 muestra el circuito.



**Figura 59.** Circuito para la tabla 29.

Por medio de estas funciones se pueden lograr las funciones de la tabla 28 al agregar el número  $N_i$  (tabla 29) a la entrada A, a través de un sumador en paralelo para cada etapa, teniendo en cuenta el valor de la entrada  $C_{in}$ . El circuito combinacional aritmético se muestra en la figura 60, la entrada A se denomina  $M_i$  en el sumador completo.

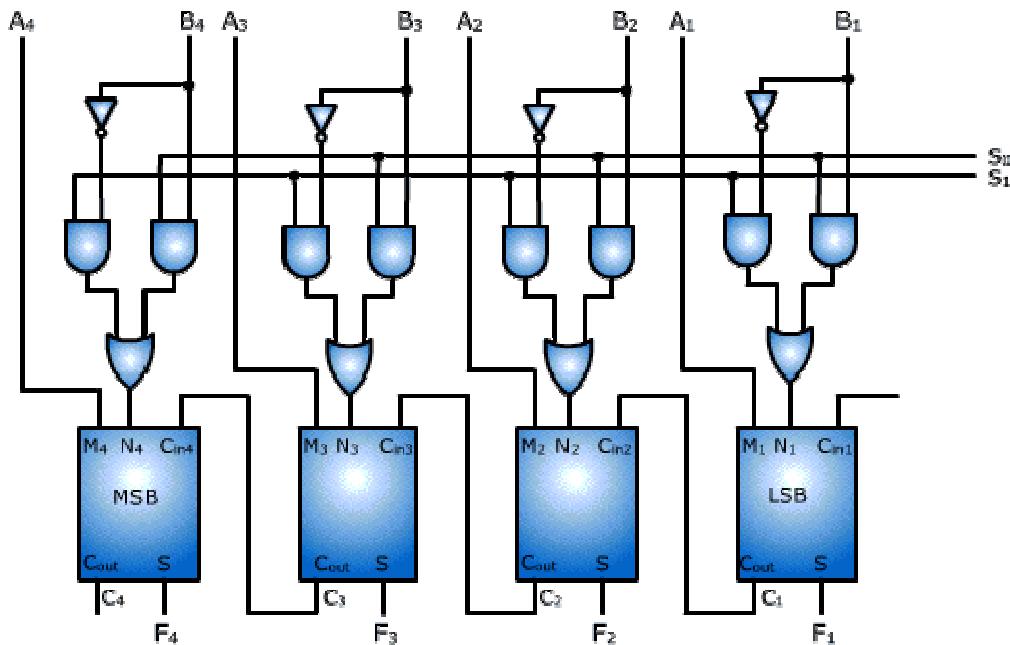


Figura 60. Circuito aritmético

**Diseño de una Unidad Aritmética Lógica:** se deben seguir los siguientes pasos:

1. Diseñar la sección aritmética independientemente de la sección lógica.
2. Determinar las operaciones lógicas del circuito aritmético, asumiendo que los acarrees de salida de todas las etapas son 0.
3. Modificar el circuito aritmético para obtener las operaciones lógicas requeridas.

El diseño simple de una ALU se hace utilizando el sumador completo para generar las operaciones lógicas de la unidad. Por lo tanto es necesario introducir una variable de control adicional ( $S_2$ ), con el fin de seleccionar entre las operaciones lógicas y aritméticas. En este diseño, un valor  $S_2 = 1$  hace que el circuito efectúe operaciones lógicas. Recordando la salida de un sumador completo:

$$F = (A_i \oplus B_i) \oplus C_{in}$$

A partir de esta ecuación, es posible obtener la función lógica requerida, utilizando la debida manipulación lógica. La función requerida se expone en la tabla 30.

$S_2$	$S_1$	$S_0$	$A_i$	$B_i$	$C_{in}$	Operación Sumador Completo	Función requerida $F_i$	Manipulación	Salida
1	0	0	$A_i$	0	0	$A_i$	OR	Aplicar una función OR $A_i + B_i$	$A_i + B_i$
1	0	1	$A_i$	$B_i$	0	$A_i \oplus B_i$	XOR	Ninguna	$A_i \oplus B_i$
1	1	0	$A_i$	$B_i'$	0	$A_i \cdot B_i$	AND	Aplicar una función OR $A_i + B_i'$	$A_i \cdot B_i$
1	1	1	$A_i$	1	0	$A_i'$	NOT	Ninguna	$A_i'$

**Tabla 30.** Tabla de obtención de las funciones lógicas con un sumador completo



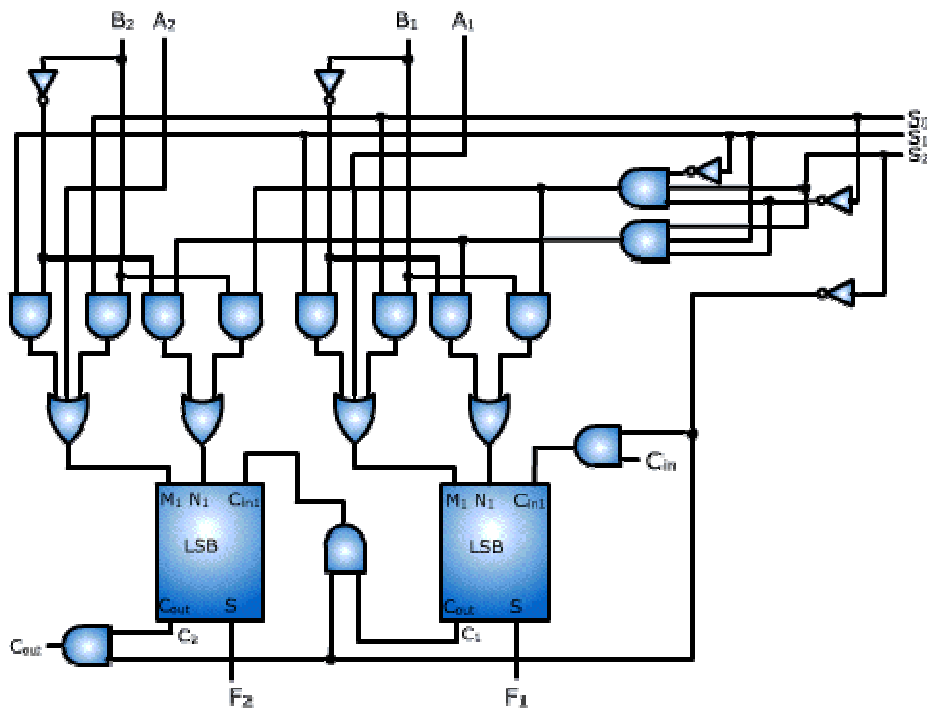
Partiendo de la tabla 30, las entradas  $M_i$ ,  $N_i$  y  $C_{ini}$  en un sumador completo, son equivalentes a las siguientes expresiones:

$$M_i = A_i + S_2 \cdot S_1' \cdot S_0' \cdot B_i + S_2 \cdot S_1 \cdot S_0' \cdot B_i'$$

$$N_i = S_0 \cdot B_i + S_1 \cdot B_i'$$

$$C_{ini} = S_2' \cdot C_i$$

La figura 61 muestra el diagrama de la unidad aritmética lógica de dos etapas.



**Figura 61.** Diagrama lógico de una ALU

Las doce operaciones generadas en el ALU se resumen en la tabla 31, la función en particular se selecciona a través de  $S_2$ ,  $S_1$ ,  $S_0$  y  $C_{in}$ . Las operaciones aritméticas son las mismas del circuito aritmético.

Selección				Salida F	Descripción
$S_2$	$S_1$	$S_0$	$C_{in}$	$F$	
0	0	0	0	A	Trasferir A
0	0	0	1	A+1	Incrementar A
0	0	1	0	A+B	Suma
0	0	1	1	A+B+1	Suma con acarreo
0	1	0	0	A-B-1	Resta con préstamo
0	1	0	1	A-B	Sustracción
0	1	1	0	A-1	Decrementar A
0	1	1	1	A	Transferir A
1	0	0	X	A+B	OR
1	0	1	X	$A \oplus B$	OR-Exclusiva
1	1	0	X	$A \cdot B$	AND
1	1	1	X	$A'$	Complementar A

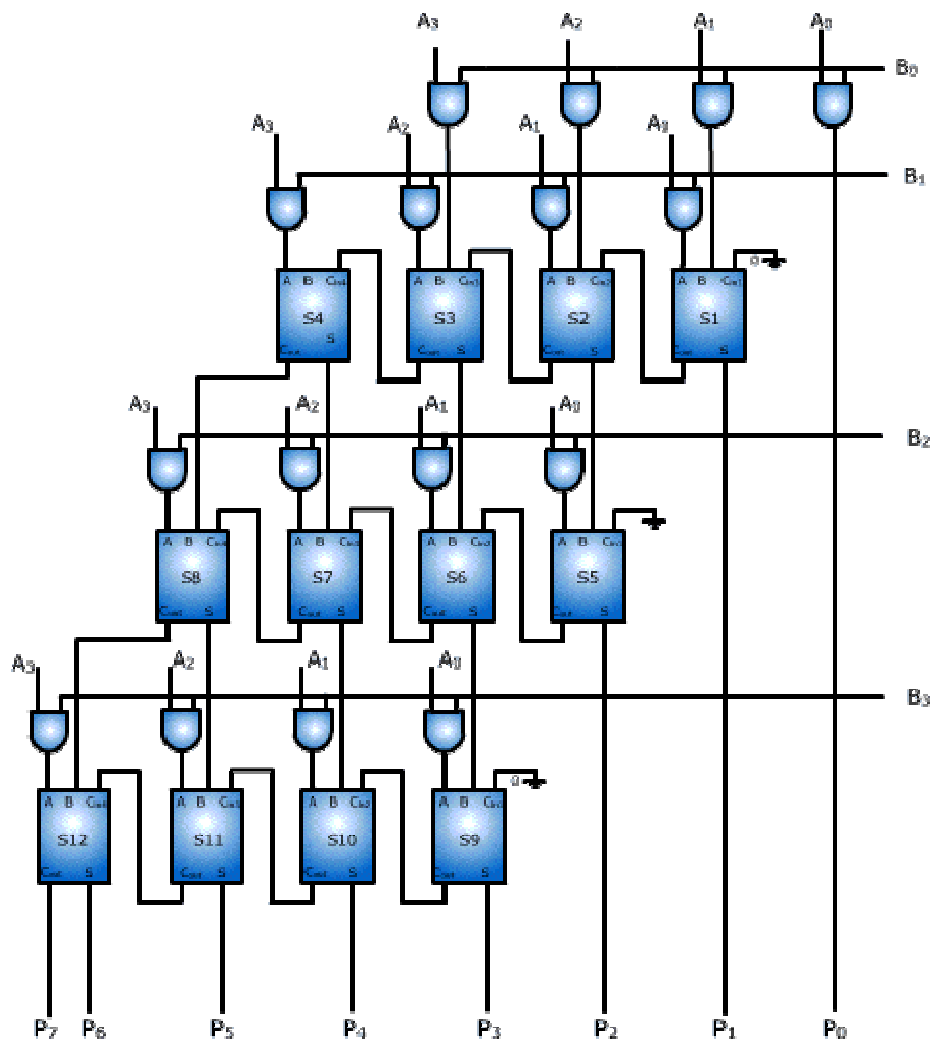
Tabla 31. Tabla de verdad de una ALU

### Multiplicador Combinatorio

Un multiplicador combinatorio permite realizar la operación de multiplicación mediante circuitos combinacionales. Como ejemplo, un circuito construido para este propósito es un multiplicador combinacional paralelo de 4 bits, mostrado en la figura 62. Este multiplicador está constituido internamente por circuitos sumadores completos, que a su vez internamente están diseñados a nivel de puertas lógicas. En el primer nivel de compuertas de la figura se obtienen las operaciones  $A_0B_0$ ,  $A_1B_0$ ,  $A_2B_0$  y  $A_3B_0$ . En el segundo nivel de compuertas, las operaciones  $A_0B_1$ ,  $A_1B_1$ ,  $A_2B_1$  y  $A_3B_1$ . En el tercero, las operaciones  $A_0B_2$ ,  $A_1B_2$ ,  $A_2B_2$  y  $A_3B_2$  y en el cuarto  $A_0B_3$ ,  $A_1B_3$ ,  $A_2B_3$  y  $A_3B_3$ . Por ejemplo,  $A_0B_0$  es directamente el resultado  $P_0$ . El dígito  $P_3$ , se obtiene de la suma de los bits de entrada a los sumadores  $S_3$ ,  $S_6$ ,  $S_9$  y el bit  $A_3B_0$ . La figura 62 recuerda el proceso de multiplicación de dos números de cuatro bits.

$$\begin{array}{r}
 \begin{array}{cccc}
 A_3 & A_2 & A_1 & A_0 \\
 B_3 & B_2 & B_1 & B_0 \\
 \hline
 B_0A_3 & B_0A_2 & B_0A_1 & B_0A_0 \\
 B_1A_3 & B_1A_2 & B_1A_1 & B_1A_0 \\
 B_2A_3 & B_2A_2 & B_2A_1 & B_2A_0 \\
 B_3A_3 & B_3A_2 & B_3A_1 & B_3A_0 \\
 \hline
 P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0
 \end{array}
 \end{array}$$

Figura 62. Multiplicación de dos números de cuatro bits

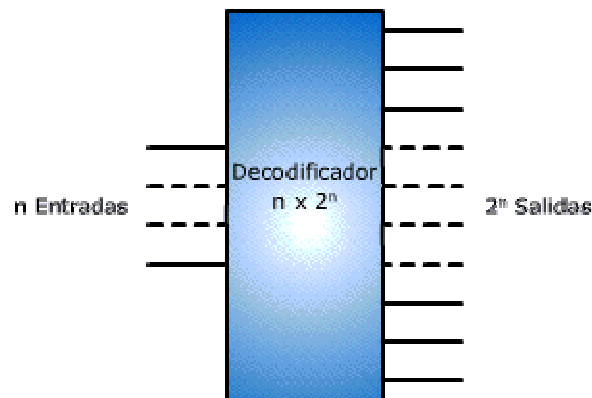


**Figura 63.** Circuito lógico del multiplicador combinatorio

### 3.2. Decodificadores:

Un decodificador es un circuito lógico cuya función es indicar la presencia de cierto código en sus líneas de entrada con un nivel predeterminado a la salida. El procedimiento consiste en interpretar el código de  $n$  líneas de entrada con el fin de activar un máximo de  $2^n$  líneas a la salida. Si el código de entrada tiene combinaciones no usadas o de no importa, la salida tendrá menos de  $2^n$  salidas. La característica predominante en los decodificadores es un mayor número de salidas con respecto al número de entradas.

$n$  Entradas -----[  $n \times 2^n$  ]-----  $2^n$  salidas



**Figura 64.** Diagrama de bloques de un Decodificador  $n \times 2^n$ .

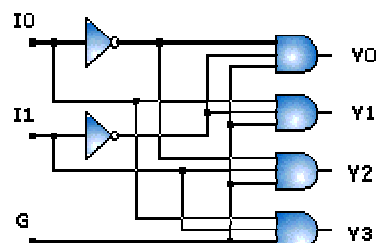
### Decodificador de 2 a 4 líneas (2 bits)

El Decodificador de 2 a 4 líneas tiene 2 líneas de entrada y 4 líneas de salida. En la tabla 32 las entradas del decodificador son  $I_0$  e  $I_1$  y representan un entero de 0 a 3 en código decimal.  $G$  es la entrada de habilitación y determina la activación del circuito de acuerdo a su valor lógico ("1" circuito activo, "0" circuito no activo). Según el valor binario presente en las 2 entradas se activa una de las 4 salidas al valor lógico 1. Por ejemplo, con el valor 1 en  $I_0$  y el valor 0 en  $I_1$  se activará la salida  $Y_1$ .

G	$I_1$	$I_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

**Tabla 32.** Tabla de verdad del Decodificador de 2 bits

En la figura 65 se muestra el circuito lógico del decodificador 2x4.



**Figura 65.** Diagrama lógico del decodificador 2 x 4 con entrada de habilitación

**Decodificador de 3 a 8 líneas (3 bits)**

El decodificador de 3 a 8 líneas activa una sola de las 8 líneas de salida de acuerdo con el código binario presente en las 3 líneas de entrada. Las salidas son mutuamente exclusivas ya que solamente una de las salidas es igual a 1 en cualquier momento.

Las entradas del decodificador son  $x$ ,  $y$ ,  $z$  y las salidas van de  $y_0$  a  $y_7$  (activas bajas). La tabla de verdad del decodificador se muestra en la tabla 33.

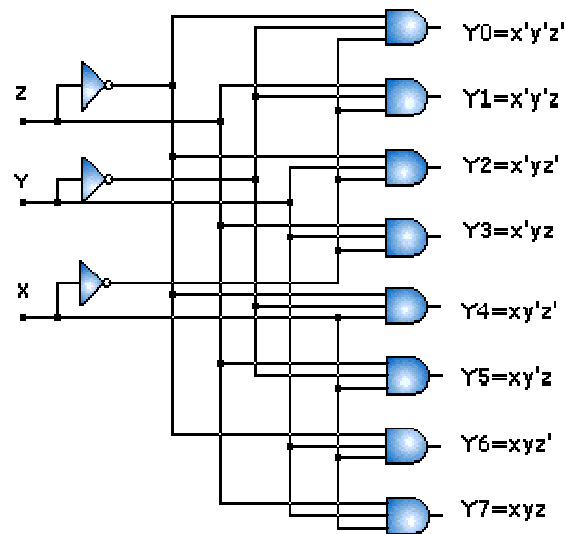
Entradas			Salidas							
X	Y	Z	$Y_0$	$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$	$Y_6$	$Y_7$
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

**Tabla 33.** Tabla de verdad para el Decodificador de 3 a 8 líneas.

Como la tabla anterior tiene 8 salidas, por lo tanto sería necesario dibujar ocho mapas de karnaugh para simplificar cada una de las funciones de salida.

Por tanto procedimiento, se puede dibujar un solo mapa y reducir la función para cada término por separado. La reducción de cada término da como resultado la equivalencia entre cada mintermino de entrada y la salida correspondiente.

Por ejemplo, la entrada 110 activará la salida  $Y_6$ . En el circuito el mintermino corresponderá a una compuerta AND de tres entradas con las variables  $A \cdot B \cdot C'$  como entradas. De manera similar se construye el circuito para el resto de entradas. El circuito lógico del decodificador de 3 a 8 líneas se representa en la figura 66.



**Figura 66.** Diagrama lógico de un Decodificador 3 x 8.

#### Decodificador de 4 a 16 líneas (4 bits)

El decodificador de 4 a 16 líneas activa una sola de las 16 líneas de salida de acuerdo con el código binario presente en las 4 líneas de entrada. Las salidas son mutuamente exclusivas ya que solamente una de las salidas es igual a 1 en cualquier momento.

Las entradas son  $w, x, y, z$  y las salidas son  $y_0$  a  $y_{15}$  (activas bajas). La tabla 34 muestra la tabla de verdad para el decodificador.

Entradas				Salidas															
w	x	y	z	$y_0$	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$	$y_7$	$y_8$	$y_9$	$y_{10}$	$y_{11}$	$y_{12}$	$y_{13}$	$y_{14}$	$y_{15}$
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

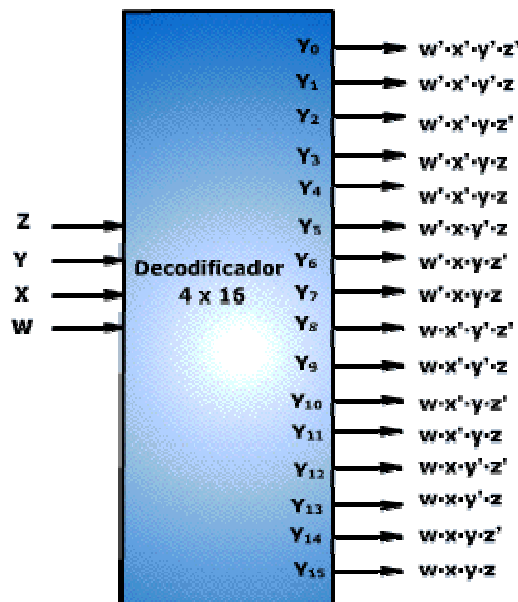
**Tabla 34.** Tabla de verdad para el decodificador de 4 a 16 líneas

Similar al decodificador de 3 a 8, la salida correspondiente a cada código es el mintermino correspondiente a cada entrada. La simplificación de la función necesitaría de 16 mapas para la reducción. En vez de construir 16 mapas, se construye solo uno, en el cuál se representa cada uno de los valores para cada combinación de entrada (Ver figura 67). Los minterminos no se pueden asociar por la consideración anterior, pero el ejemplo sirve para mostrar la construcción del circuito lógico.

ZW XY	ZW			
	00	01	11	10
00	$Y_0$	$Y_1$	$Y_3$	$Y_2$
01	$Y_4$	$Y_5$	$Y_7$	$Y_6$
11	$Y_{12}$	$Y_{13}$	$Y_{15}$	$Y_{14}$
10	$Y_8$	$Y_9$	$Y_{11}$	$Y_{10}$

**Figura 67.** Mapa de karnaugh de la función del decodificador de 4 a 16 líneas

En la tabla el término  $Y_7$  se obtiene del mintermino  $m_7 (W' \cdot Z \cdot Y \cdot X)$ . En la entrada, los valores 0111 activarán la salida  $Y_7$ . El resto del circuito lógico se construye de manera similar. El diagrama de bloques del circuito lógico se representa en la figura 68.



**Figura 68.** Diagrama de bloques del decodificador 4 a 16 líneas

### Ejemplos de Aplicación en los Computadores

En la comunicación entre los diferentes dispositivos que conforman un computador, se emplean puertos de E/S y memorias. Entre las aplicaciones más comunes de los decodificadores se encuentra la habilitación de puertos de E/S en los computadores.

Cada uno de los dispositivos dentro de un computador posee una dirección que es codificada mediante un código binario (dirección) y cuando es necesario comunicarse con un dispositivo, la *CPU* del computador envía la dirección del puerto o posición de memoria al que se encuentra conectado el dispositivo. El código binario de la dirección es decodificado, activando la salida que habilita el dispositivo correspondiente.

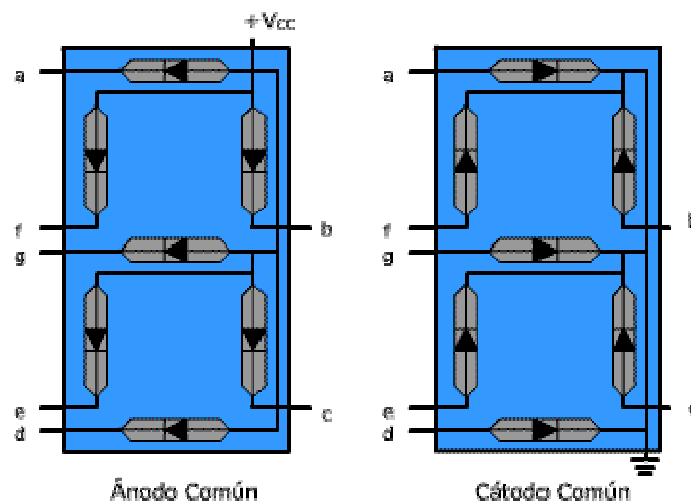
Los decodificadores también son utilizados internamente en los chips de memoria para direccionar las posiciones de memoria de las palabras binarias almacenadas. Como ejemplo, un computador que maneja direcciones de 16 *bits*, tiene la capacidad de direccionar  $2^{16} = 65536$  posiciones de memoria, o lo que equivale a 64 K.

### Decodificadores BCD a 7 segmentos

El decodificador de *BCD* a siete segmentos es un circuito combinacional que permite un código *BCD* en sus entradas y en sus salidas activa un *display* de 7 segmentos para indicar un dígito decimal.

### El Display de Siete Segmentos

El *display* está formado por un conjunto de 7 leds conectados en un punto común en su salida. Cuando la salida es común en los ánodos, el *display* es llamado de ánodo común y por el contrario, si la salida es común en los cátodos, llamamos al *display* de cátodo común. En la figura 69, se muestran ambos tipos de dispositivos. En el *display* de cátodo común, una señal alta encenderá el segmento excitado por la señal. La alimentación de cierta combinación de *leds*, dará una imagen visual de un dígito de 0 a 9.

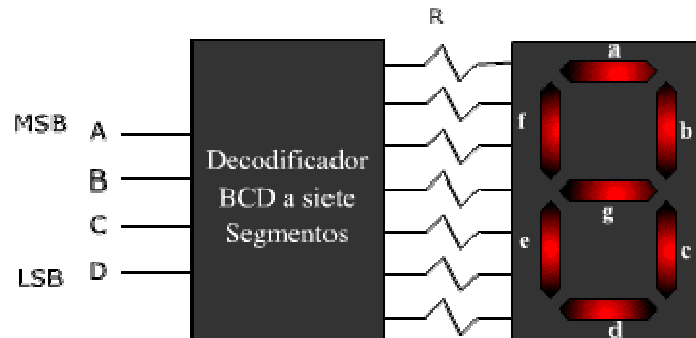


**Figura 69.** Display de ánodo común y cátodo común



### Decodificador de BCD a Siete Segmentos

El decodificador requiere de una entrada en código decimal binario *BCD* y siete salidas conectadas a cada segmento del *display*. La figura 70 representa en un diagrama de bloques el decodificador de *BCD* a 7 segmentos con un display de cátodo común.



**Figura 70.** Diagrama de bloques de un decodificador BCD a siete segmentos

Suponiendo que el visualizador es un display de cátodo común, se obtiene una tabla cuyas entradas en código *BCD* corresponden a *A*, *B*, *C* y *D* y unas salidas correspondientes a los leds que se encenderían en cada caso para indicar el dígito decimal. La tabla 35 muestra el caso de ejemplo.

Valor decimal	Entradas				Salidas						
	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
...	..	..	..	..	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

**Tabla 35.** Tabla de verdad del decodificador BCD a siete segmentos.

Los valores binarios *1010* a *1111* en *BCD* nunca se presentan, entonces las salidas se tratan como condiciones de no importa.

La simplificación de la información contenida en la tabla 35 requiere de siete tablas de verdad, que se pueden separar para cada segmento. Por consiguiente, un 1 en la columna indica la activación del segmento y varios de estos segmentos activados indican visualmente el número decimal requerido.

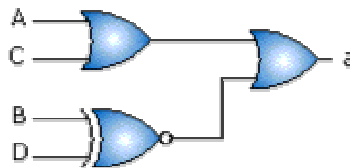
Según la información de la tabla de verdad, se puede obtener la expresión para cada segmento en suma de productos o producto de sumas según la cantidad de unos y ceros presentes.

Salida a En la columna a existen 3 ceros y 7 unos, entonces es más fácil obtener la función PDS:

$$a = (A+B+C+D') \cdot (A+B'+C+D) = A + D \cdot (B+C) + B' \cdot (D'+C) = A + A \cdot B' + A \cdot C + A \cdot D + B \cdot A + B \cdot C + B \cdot D + C \cdot A + C \cdot B' + C + C \cdot D + D' \cdot A + D' \cdot B' + D' \cdot C$$

$$a = A + (A \cdot B' + B \cdot A) + (A \cdot C + C \cdot A) + (A \cdot D + D' \cdot A) + (B \cdot C + C \cdot B') + B \cdot D + C + (C \cdot D + D' \cdot C) + D' \cdot B' = A + A + A \cdot C + A + C + B \cdot D + C + C + D' \cdot B' = A + A \cdot C + C + B \cdot D + D' \cdot B'$$

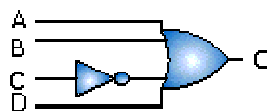
$$a = A + C + (B \oplus D)'$$



**Figura 71.** Circuito para la salida a del decodificador BCD a siete segmentos

Salida c En la columna de la salida c se tiene un solo 0, entonces se emplea el PDS:

$$c = (A + B + C' + D)$$

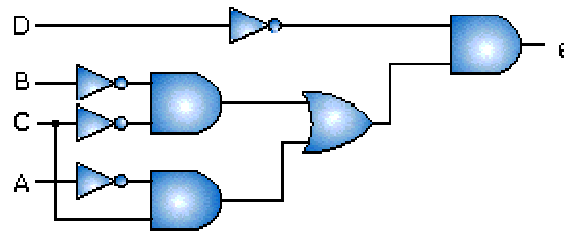


**Figura 72.** Circuito para la salida c del decodificador BCD a siete segmentos

Salida e La columna correspondiente a esta salida tiene 4 unos y 5 ceros. Es mejor utilizar la representación SDP:

$e = (A' \cdot B' \cdot C' \cdot D') + (A' \cdot B' \cdot C \cdot D') + (A' \cdot B \cdot C \cdot D') + (A \cdot B' \cdot C' \cdot D')$ ; factorizando el primer término con el cuarto y el segundo con el tercero:

$$e = B' \cdot C' \cdot D' + A' \cdot C \cdot D' = D' \cdot (B' \cdot C' + A' \cdot C)$$

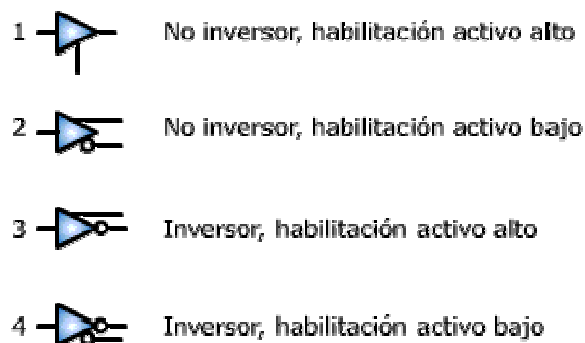


**Figura 73.** Circuito para la salida e del decodificador BCD a siete segmentos

El resto de salidas se obtiene por las mismas deducciones anteriores.

### 3.3. Registros de Tres Estados

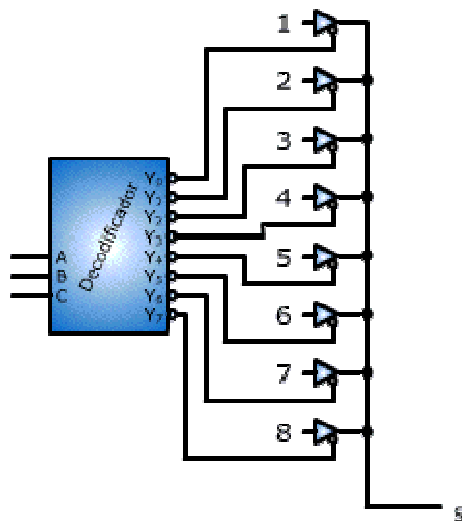
El principio básico de un registro de estados es la presencia de tres estados para la salida del dispositivo (0, 1 y alta impedancia) según el valor de una entrada de control predeterminada. El dispositivo más básico es el registro ("buffer") de tres estados. Este registro posee una entrada de habilitación ("*entrada lateral al registro*") para determinar su comportamiento como amplificador, inversor ordinario o dispositivo de alta impedancia. La figura 74 muestra el símbolo lógico del registro. En los casos 1 y 3 se habilita con estado activo alto y en los casos 2 y 4 se habilita con estado activo bajo. En estado de activación la salida se comporta como amplificador o inversor. Cuando la entrada de habilitación se niega, la salida va a un estado de alta impedancia (Z).



**Figura 74.** Registros de tres estados

Estos dispositivos permiten que varias fuentes puedan compartir una misma línea de comunicación, siempre y cuando una sola fuente utilice la línea a la vez. Un circuito de este tipo se muestra en la figura 75. El circuito se configura con un decodificador para seleccionar una de ocho líneas de salida.

Por ejemplo, la selección 001 habilita la salida  $Y_1$  en estado bajo, activando el registro 2 y coloca la información de entrada del registro en la línea de comunicación.



**Figura 75.** Circuito lógico para una línea de comunicación

Los registros de tres estados pasan más rápidamente al estado Z. Por el contrario, el tiempo de transición para salir del estado Z es mucho más demorado. El tiempo muerto en la línea de comunicación debe ser lo bastante largo para tomar en cuenta las diferencias del peor caso entre los tiempos de activación y desactivación de los dispositivos al igual que las asimetrías en las señales de control de los tres estados.

### 3.4. Codificadores y Decodificadores

**Un codificador** tiene  $2^n$  o menos líneas de entrada y  $n$  líneas de salida. Por ejemplo, en una de las entradas se puede ingresar un dígito decimal u octal y generarse un código de salida en BCD o binario. La función de los codificadores es inversa a la de los decodificadores. Los codificadores se utilizan también para codificar símbolos diferentes y caracteres alfabéticos.

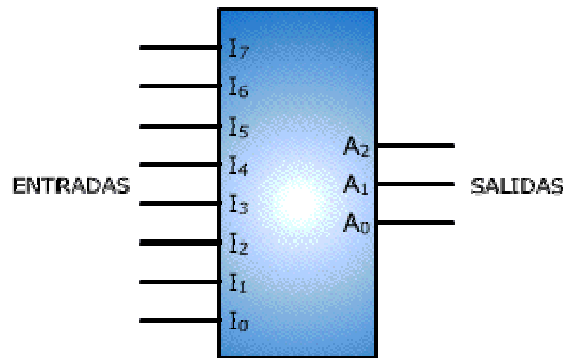
$2^n$  Entradas -----[                      ]----- n salidas

#### Codificador Binario

El codificador binario tiene  $2^n$  entradas y  $n$  salidas. Sólo, una sola de las entradas puede estar activada. La salida suministra el valor binario correspondiente a la entrada activada. Este tipo de decodificador opera en forma contraria a los decodificadores de 2 a 4, 3 a 8, estudiados antes.

#### Codificador de 8 a 3.

El codificador 8 a 3 tiene 8 entradas ( $I_0$  a  $I_7$ ), una para cada uno de los ocho dígitos y 3 salidas que conforman el número binario equivalente ( $A_0$  a  $A_2$ ). La figura 76 muestra en el diagrama de bloques del decodificador.

**Figura 76.** Codificador de 8 a 3

La tabla de verdad se muestra en la tabla 36.

Entradas								Salidas		
$I_0$	$I_1$	$I_2$	$I_3$	$I_4$	$I_5$	$I_6$	$I_7$	$A_2$	$A_1$	$A_0$
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

**Tabla 36.** Tabla de verdad de codificador de 8 a 3.

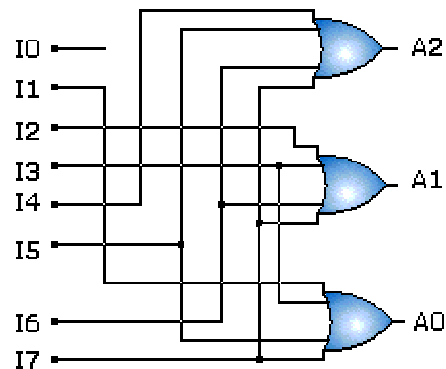
En la tabla de verdad,  $A_0$  tiene un 1 lógico para la columnas de entrada con subíndice impar. La salida  $A_1$  es 1 en la columnas  $I_2$ ,  $I_3$ ,  $I_6$  e  $I_7$  y la salida  $A_2$  es 1 en la columnas  $I_4$ ,  $I_5$ ,  $I_6$  e  $I_7$ . Las expresiones lógicas son las siguientes:

$$A_0 = I_1 + I_3 + I_5 + I_7$$

$$A_1 = I_2 + I_3 + I_6 + I_7$$

$$A_2 = I_4 + I_5 + I_6 + I_7$$

Por ejemplo, si está activada la entrada 3, la salida es 011. El circuito se construye con compuertas OR y se muestra en la figura 77.



**Figura 77.** Circuito lógico del decodificador 8 a 3.

### Codificador sin prioridad

Los circuitos codificadores pueden ser diseñados con prioridad o sin ella. En los codificadores sin prioridad con entradas activas altas, la activación de más de una entrada simultáneamente con valor 1, genera un código erróneo en la salida, de acuerdo al número de entradas excitadas con el respectivo valor. La solución de este conveniente se logra empleando codificadores de prioridad.

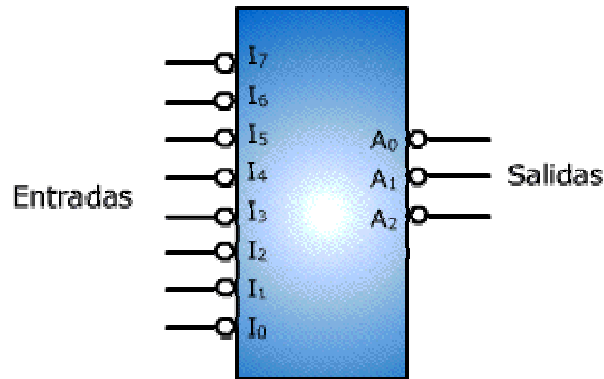
### Codificador de prioridad

Los codificadores de prioridad seleccionan la entrada de mayor prioridad cuando se presentan varias entradas activas simultáneamente. En la tabla 37 se muestra la lógica de entrada y de salida de un decodificador.

Entradas								Salidas		
$I_0$	$I_1$	$I_2$	$I_3$	$I_4$	$I_5$	$I_6$	$I_7$	$A_2$	$A_1$	$A_0$
X	X	X	X	X	X	X	0	0	0	0
X	X	X	X	X	X	0	1	0	0	1
X	X	X	X	X	0	1	1	0	1	0
X	X	X	X	0	1	1	1	0	1	1
X	X	X	0	1	1	1	1	1	0	0
X	X	0	1	1	1	1	1	1	0	1
X	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1

**Tabla 37.** Tabla de verdad del Codificador de Prioridad.

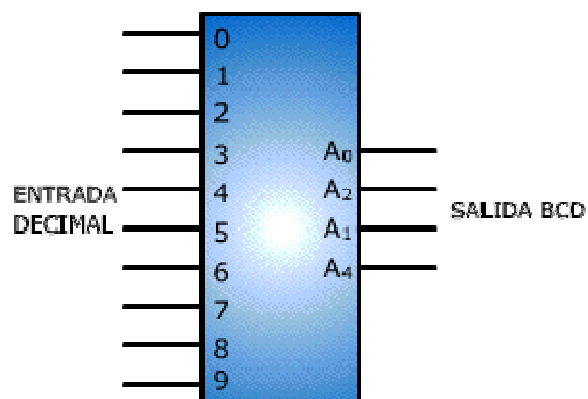
El **decodificador** se encuentra comercialmente tal como se encuentra dispuesto en la figura 78. La diferencia radica en unas entradas de habilitación adicionales que activan las entradas ó las salidas a unos valores predefinidos.



**Figura 78.** Diagrama de Bloques del codificador de Prioridad.

#### Codificador Decimal - BCD

El codificador decimal a *BCD* posee diez entradas, correspondientes cada una a un dígito decimal y cuatro salidas en código *BCD* (8421). El diagrama de bloques de la figura 79 muestra la disposición de entradas y salidas del decodificador.



**Figura 79.** Diagrama de Bloques del codificador Decimal a BCD.

En la tabla 38 se encuentra el código *BCD* correspondiente a cada dígito decimal.

Dígito Decimal	BCD			
	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

**Tabla 38.** Código Decimal –BCD.

El bit  $A_3$  es el más significativo del código *BCD* y es 1 para los decimales 8 ó 9. La expresión para este bit en función de los dígitos decimales se escribe:

$$A_3 = 8 + 9$$

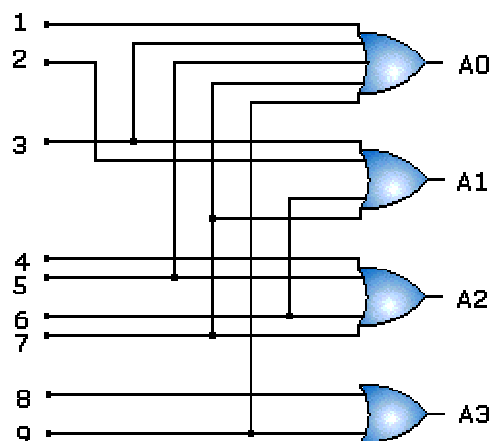
Por tanto las funciones siguientes corresponden a:

$$A_2 = 4 + 5 + 6 + 7$$

$$A_1 = 2 + 3 + 6 + 7$$

$$A_0 = 1 + 3 + 5 + 7 + 9.$$

Ahora configurando el análisis en un circuito combinacional, se obtiene el siguiente circuito sin necesidad de una entrada para el bit 0.

**Figura 80.** Circuito lógico del codificador BCD a Decimal



## Aplicaciones

Los codificadores encuentran mayor aplicación en los dispositivos de entrada y salida. La señal de entrada es introducida de una forma comprensible para el usuario y la "traducción" la realiza el codificador a un código comprensible para el equipo. En un teclado, cuando se pulsa la tecla correspondiente a un dígito, esta entrada se codifica en código *BCD*.

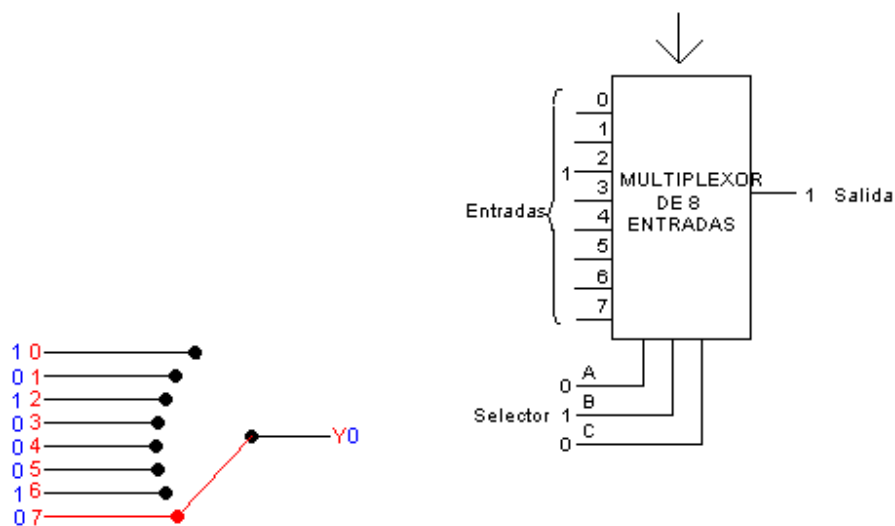
### 3.5. Multiplexores y Demultiplexores

Multiplexar es transmitir datos de una de  $n$  fuentes a la salida del circuito combinacional. El demultiplexor desempeña la función contraria.

#### Multiplexores (MUX) Selector de Datos

Es la versión electrónica de un conmutador rotatorio en un solo sentido, se puede comparar con un selector mecánico en una sola dirección. También se puede definir como un proceso de selección de una entrada entre varias y la transmisión de los datos seleccionados hacia un solo canal de salida.

Un multiplexor es un circuito combinacional que selecciona una de  $n$  líneas de entrada y transmite su información binaria a la salida. La selección de la entrada es controlada por un conjunto de líneas de selección. La relación de líneas de entrada y líneas de selección está dada por la expresión  $2^n$ , donde  $n$  corresponde al número de líneas de selección y  $2^n$  al número de líneas de entrada.

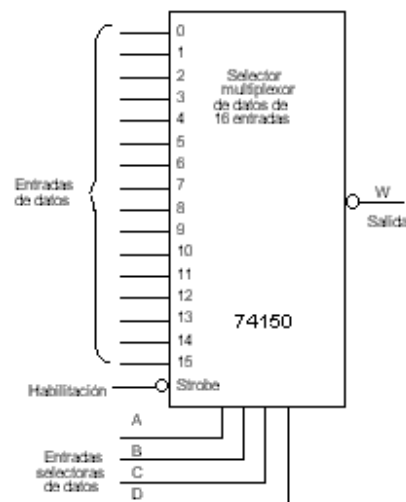


**Figura 81.** Multiplexores (MUX) Selector de Datos

En la figura 81, se compara un selector mecánico de datos y un selector electrónico de datos. En el primer caso la selección del dato se logra girando mecánicamente el rotor del conmutador, y en el selector electrónico de datos multiplexor se selecciona el dato colocando el número binario adecuado en las entradas de selección de datos A, B, C.

A continuación se ilustra el multiplexor comercial TTL 74150 que tiene las siguientes características:

1. Consta de 16 entradas de datos.
2. Tiene una única salida invertida w (pin 10).
3. Posee cuatro entradas selectoras de datos de A a D (pin 15 al 11).
4. Tiene una entrada de habilitación denominada STROBE que se considera como un conmutador ON-OFF.



**Figura 82.** Multiplexores (MUX) Selector de Datos 74150

La tabla de verdad del selector de datos 74150 nos muestra en su primera línea la entrada de habilitación (STROBE) en alto lo cual no habilita ningún dato, sea cualquiera la entrada de selección, como resultado obtendremos en la salida una tensión alta. En la segunda línea tenemos las entradas de habilitación en bajo lo cual habilita las entradas selectoras de datos que en este caso están en bajo por lo cual en la salida obtendremos la entrada E.

### Multiplexor de 2 entradas

El multiplexor se caracteriza por tener dos líneas de entrada, una línea de selección y una de salida. En el multiplexor, las entradas son  $I_0$  e  $I_1$  y la selección viene dada por el valor de la entrada S.

El valor de la salida Y depende de los valores lógicos ingresados en los cuadros de texto para las variables  $I_0$ ,  $I_1$  y S. Por ejemplo, si  $I_0=0$ ,  $I_1=1$  y  $S=0$ , entonces  $Y=I_0=0$ .

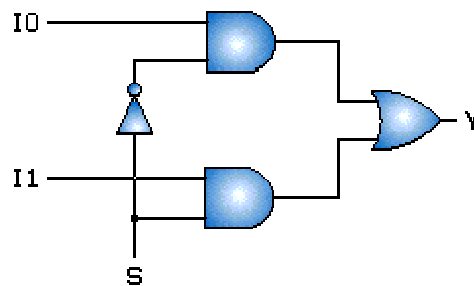
La tabla de verdad se muestra en la tabla 39.



0	$I_0$
1	$I_1$

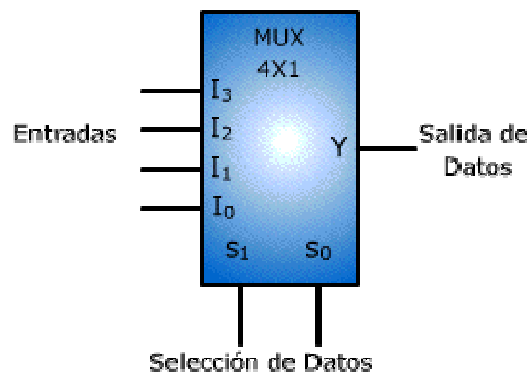
**Tabla 39.** Tabla de verdad de un multiplexor de dos entradas

El circuito lógico se muestra en la figura 83.

**Figura 83.** Multiplexor 2 a 1

### Multiplexor de 4 entradas

El multiplexor de 4 entradas es un multiplexor de 4 líneas a 1. La figura 84 muestra el diagrama de bloques del multiplexor. Las entradas son  $I_0$ ,  $I_1$ ,  $I_2$  e  $I_3$  y la selección viene dada por las entradas  $S_0$  y  $S_1$ . El valor de la salida  $Y$  depende de los valores lógicos presentes en las entradas de datos y la selección.

**Figura 84.** Multiplexor 4 a 1

La tabla de verdad se muestra en la tabla 40. Por ejemplo, si  $I_0=1$ ,  $I_1=1$ ,  $I_2=0$ ,  $I_3=1$  y  $S_1=1$ ,  $S_0=0$  entonces  $Y=I_2=0$ .

Entrada de Selección de datos		Entrada Seleccionada
$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

**Tabla 40.** Tabla de verdad de un multiplexor de cuatro entradas.

El problema consiste en definir un conjunto de expresiones para construir el circuito lógico. La ecuación en cada fila, se obtiene a partir del dato de entrada y la entrada de selección de datos:

La salida es  $Y = I_0$ , si  $S_1 = 0$  y  $S_0 = 0$ . Entonces  $Y = I_0 \cdot S_1' \cdot S_0'$ .

La salida es  $Y = I_1$ , si  $S_1 = 0$  y  $S_0 = 1$ . Entonces  $Y = I_1 \cdot S_1' \cdot S_0$ .

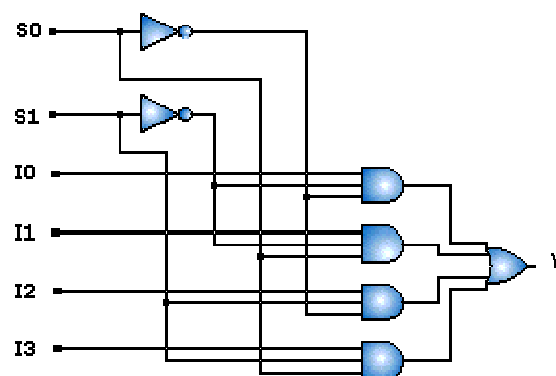
La salida es  $Y = I_2$ , si  $S_1 = 1$  y  $S_0 = 0$ . Entonces  $Y = I_2 \cdot S_1 \cdot S_0'$ .

La salida es  $Y = I_3$ , si  $S_1 = 1$  y  $S_0 = 1$ . Entonces  $Y = I_3 \cdot S_1 \cdot S_0$ .

Sumando lógicamente las ecuaciones anteriores:

$$Y = I_0 \cdot S_1' \cdot S_0' + I_1 \cdot S_1' \cdot S_0 + I_2 \cdot S_1 \cdot S_0' + I_3 \cdot S_1 \cdot S_0$$

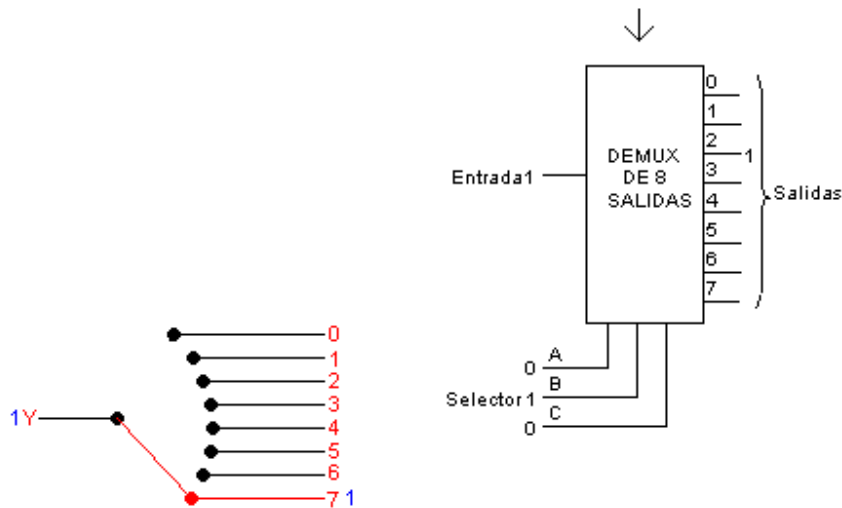
En consecuencia, el circuito asociado se implementa en la figura 85.



**Figura 85.** Circuito Lógico de un multiplexor 4 a 1

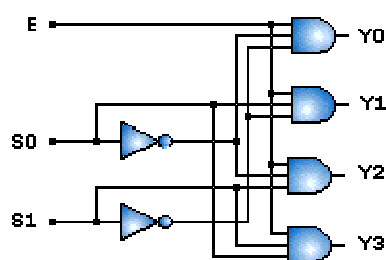
### Demultiplexores DEMUX (Distribuidores de datos)

Un demultiplexor es un circuito combinacional que recibe información en una sola línea y la transmite a una de  $2^n$  líneas posibles de salida. La selección de una línea de salida específica se controla por medio de los valores de los bits de  $n$  líneas de selección. La operación es contraria al multiplexor.



**Figura 86.** Demultiplexor (Distribuidor de datos)

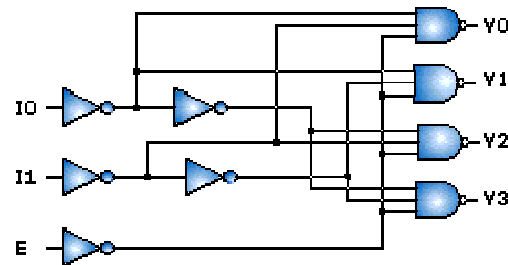
La figura 87 muestra un demultiplexor de 1 a 4 líneas. Las líneas de selección de datos activan una compuerta cada vez y los datos de la entrada pueden pasar por la compuerta hasta la salida de datos determinada. La entrada de datos se encuentra en común a todas las AND.



**Figura 87.** Circuito Lógico de un Demultiplexor de 1 a 4 líneas.

El decodificador de la figura 88 funciona como un demultiplexor si la línea  $E$  se toma como línea de entrada de datos y las líneas  $I_0$  e  $I_1$  como líneas de selección. Observe que la variable de entrada  $E$  tiene un camino a todas las salidas, pero la información de entrada se dirige solamente a una de las líneas de salida de acuerdo al valor binario de las dos líneas de selección  $I_0$  e  $I_1$ . Por ejemplo si la selección de las líneas  $I_0 I_1 = 10$  la salida  $Y_2$

tendrá el mismo valor que la entrada  $E$ , mientras que las otras salidas se mantienen en nivel bajo.



**Figura 88.** Circuito Lógico de un Decodificador/Demultiplexor.

En consecuencia, como las operaciones decodificador y demultiplexor se obtienen del mismo circuito, un decodificador con una entrada de activación se denomina *decodificador/demultiplexor*; siendo la entrada de activación la que hace al circuito un demultiplexor.

La tabla de verdad se muestra en la tabla 41

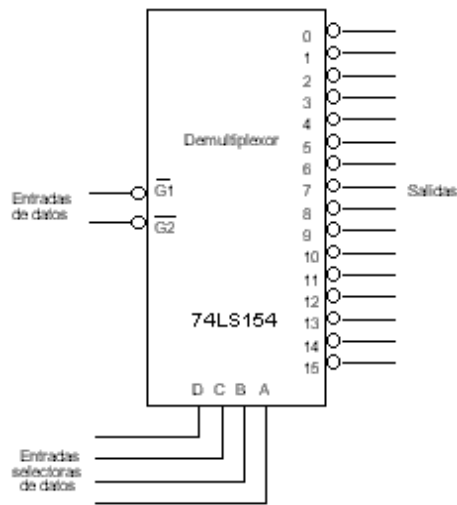
E	I <sub>0</sub>	I <sub>1</sub>	Y <sub>0</sub>	Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>3</sub>
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

**Tabla 41.** Tabla de verdad de un decodificador/demultiplexor

Los DEMUX están disponibles en versiones TTL y CMOS de una entrada y cuatro salidas, una entrada y ocho salidas, una entrada y diez salidas y una entrada y dieciséis salidas.

El CI decodificador/demultiplexor de 4 a 16 TTL 74LS154 tiene dos entradas de datos G1 y G2 que activan a una única entrada en el nivel BAJO.

La figura 89 muestra el DEMUX 74LS154 que tiene 16 salidas de 0 a 15 con 4 entradas de datos (D a A) sus salidas son activas en bajo por lo que normalmente están en alto y cuando se activan están en bajo, además como se había dicho antes tiene dos entradas de datos G1 y G2 negados que realizan la operación NOR para generar la única entrada de datos lo que quiere decir que para poder activar un dato deben estar los dos en bajo.



**Figura 89.** Demultiplexor 74154.

### 3.6. Generadores de Paridad

La transmisión binaria por diversos medios de comunicación está sujeta a errores por fallas en los sistemas digitales o la presencia de ruido eléctrico. Cualquier condición interna o externa al sistema puede alterar el valor de los ceros a unos o viceversa. Cuando se altera un solo bit, decimos que el bit distorsionado contiene un error individual.

De la misma forma, dos o más bits distorsionados, involucran un error múltiple, pero estos errores tienen menor probabilidad de ocurrencia a los errores individuales. Un código que permite detectar errores es el código de paridad. El principio es añadir un bit de paridad para hacer que el número total de bits (incluida la palabra) sea par o impar. Un bit de paridad par, incluido con el mensaje (palabra), convierte el número total de unos en par (paridad par) y el bit de paridad impar hace el total de unos impar (paridad impar).

El generador de paridad es un sistema combinacional que permite generar el bit de paridad de una palabra de código. La información se transmite y el comprobador de paridad recibe la información con el fin de validarla.

*Ejemplo* Construir un generador de paridad par y el respectivo comprobador de paridad para tres bits .

En la tabla 42 los bits de entrada  $A$ ,  $B$ ,  $C$  constituyen el mensaje y el bit de paridad  $P$  la salida. En la tabla, se escoge  $P$  de tal forma que la suma de todos los unos es par.

Mensaje de tres Bits			Bit de paridad Par generado
A	B	C	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

**Tabla 42.** Tabla de verdad de un generador de paridad.

La figura 90 muestra la función en un mapa de karnaugh de tres variables.

		BC			
		00	01	11	10
A	0	0	1	0	1
	1	1	0	1	0

**Figura 90.** Mapa de Karnaugh del generador de paridad

La paridad esta directamente relacionada con la operación EXOR. En una expresión *OR-Exclusiva* de n variables,  $2^n/2$  términos mínimos tienen un número par de unos. La otra mitad tiene un número impar de unos. Observando el mapa se puede deducir que la mitad de los términos mínimos tiene un número par de unos. La función puede expresarse en términos de una operación EXOR con las tres variables de la siguiente forma:

$$P = \Sigma (m_1, m_2, m_4, m_7)$$

*Asumiendo*

$$P = \Sigma (m_1, m_2, m_4, m_7) = (A \oplus B) \oplus C$$

$$= (A \cdot B' + A' \cdot B) \oplus C$$

$$= (A \cdot B' + A' \cdot B) \cdot C' + (A \cdot B' + A' \cdot B)' \cdot C$$

$$= A \cdot B' \cdot C' + A' \cdot B \cdot C' + [(A \cdot B')' \cdot (A' \cdot B)'] \cdot C$$



$$\begin{aligned}
&= A \cdot B' \cdot C' + A' \cdot B \cdot C' + [(A' + B) \cdot (A + B')] \cdot C \\
&= A \cdot B' \cdot C' + A' \cdot B \cdot C' + (A' \cdot A + A' \cdot B' + B \cdot A + B \cdot B') \cdot C \\
&= A \cdot B' \cdot C' + A' \cdot B \cdot C' + A' \cdot B' \cdot C + A \cdot B \cdot C
\end{aligned}$$

Llegamos a la igualdad,

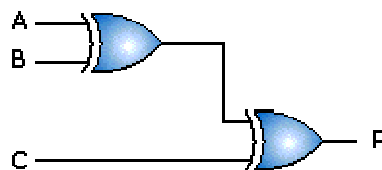
$$P = \Sigma (m_1, m_2, m_4, m_7) = A' \cdot B' \cdot C + A' \cdot B \cdot C' + A \cdot B' \cdot C' + A \cdot B \cdot C$$

Entonces,

$$P = A \oplus B \oplus C$$

El circuito realiza la función *EXOR* de un número  $n$  de variables, constituyendo a la salida un uno lógico si el número de unos aplicados a sus entradas es impar y un cero si el número es par.

El diagrama lógico del generador de paridad se muestra en la figura 91. El circuito está conformado por dos compuertas *EXOR*.



**Figura 91.** Circuito Lógico para el Generador de Paridad Par de tres bits.

El bit de paridad y el mensaje de tres bits, se transmiten a su destino donde se aplican a un circuito de observación de paridad.

La salida C del comprobador de paridad debe ser 1 para indicar el error de transmisión. El error se presenta cuando el número de unos en sus entradas es impar.

La tabla de verdad 43 muestra las entradas y las salidas del circuito.

Bits de entrada				Comprobación del Error
A	B	C	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

**Tabla 43.** Mapa de Karnaugh del comprobador de paridad.

La figura 92 muestra la función en un mapa de karnaugh de tres variables.

AB \ CP				
	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

**Figura 92.** Mapa de Karnaugh del comprobador de paridad.

En el mapa de karnaugh se pueden observar los unos en los mintérminos que tienen un número impar de unos. La función puede expresarse en términos de la operación *OR-Exclusiva*. La demostración es la siguiente:

$$CP = A \oplus B \oplus C \oplus D$$

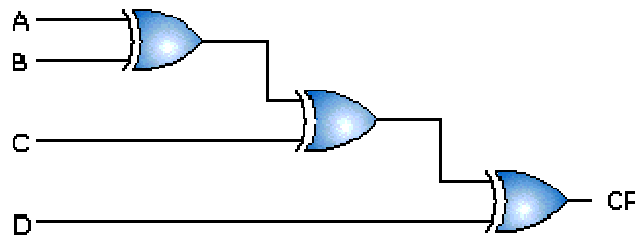
$$= A \oplus B \oplus C \oplus D$$

$$\begin{aligned}
 &= (A \oplus B) \oplus (C \oplus D) \\
 &= (A \cdot B' + A' \cdot B) \oplus (C \cdot D' + C' \cdot D) \\
 &= (A \cdot B' + A' \cdot B) \cdot (C \cdot D + C' \cdot D') + (A \cdot B + A' \cdot B') \cdot (C \cdot D' + C' \cdot D)
 \end{aligned}$$

Entonces,

$$CP = \Sigma (m_1, m_2, m_4, m_7, m_8, m_{11}, m_{13}, m_{14}).$$

El circuito lógico se muestra en la figura 93



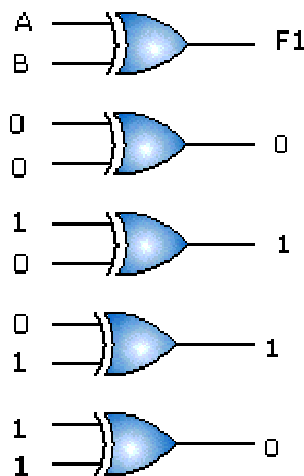
**Figura 93.** Circuito Lógico para el comprobador de paridad par de tres bits

### 3.7. Comparadores

Los circuitos comparadores son sistemas combinacionales que comparan la magnitud de dos números binarios de  $n$  bits e indican cuál de ellos es mayor, menor o si existe igualdad entre ellos. Existen varias configuraciones de circuitos de un nivel sencillo a uno más complejo para determinar relaciones de magnitud.

#### Comparador de Magnitudes de un Bit

La comparación de dos bits se puede realizar por medio de una compuerta EXOR o una NEXOR. La salida del circuito es 1 si sus dos bits de entrada son diferentes y 0 si son iguales. La figura 94. muestra el circuito comparador de magnitudes de un bit.



**Figura 94.** Comparador de magnitudes de un bit

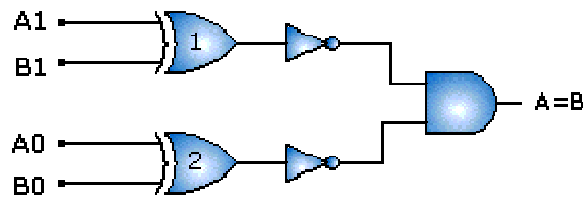
#### Comparador de Magnitudes de Dos Bits

Los números  $A$  y  $B$  de dos bits en orden significativo ascendente a descendente se ordenan de la siguiente forma:

$$A = A_1 \cdot A_0$$

$$B = B_1 \cdot B_0$$

En un comparador de dos bits se utilizan dos compuertas EXOR. El comparador se muestra en la figura 95. Los bits más significativos se comparan en la compuerta 1 y los dos menos significativos en la compuerta 2. En el caso de números iguales, los bits también son iguales, teniendo como salida en cada EXOR el valor 0. Cada EXOR se invierte y la salida de la compuerta AND tendrá un 1. En números diferentes, los bits serán diferentes y la salida de cada EXOR será 1.



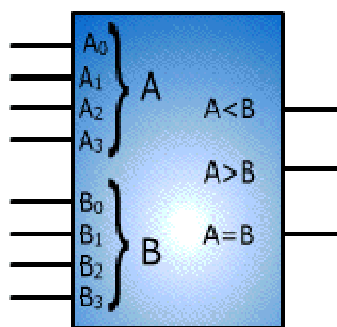
**Figura 95.** Comparador de magnitudes de dos bits.

### Comparador de magnitudes de cuatro bits

En el diagrama 96 se muestra un comparador de magnitud de cuatro bits. Las entradas son  $A$  y  $B$  y las salidas son las tres variables binarias  $A > B$ ,  $A = B$  y  $A < B$ . Escribiendo los coeficientes de los números  $A$  y  $B$  en orden significativo de ascendente a descendente:

$$A = A_3 \cdot A_2 \cdot A_1 \cdot A_0 = A_{i+3} \cdot A_{i+2} \cdot A_{i+1} \cdot A_i$$

$$B = B_3 \cdot B_2 \cdot B_1 \cdot B_0 = B_{i+3} \cdot B_{i+2} \cdot B_{i+1} \cdot B_i$$



**Figura 96.** Comparador de magnitudes de cuatro bits.

*Salida  $A=B$*

Los dos números son iguales si todos los números del mismo peso son iguales, es decir  $A_3=B_3$ ,  $A_2=B_2$ ,  $A_1=B_1$  y  $A_0=B_0$ .

La igualdad de los números  $A_i$  y  $B_i$  se determina comparando los coeficientes según el valor 0 ó 1 para los dos bits. En la comparación se emplea la variable  $y_i$ . Esta variable binaria es igual a 1 si los números de entrada  $A$  y  $B$  son iguales, de lo contrario será igual a 0. Por consiguiente, la comparación de dos bits en la posición  $i$  de un número, está dada por:

$$y_i(A_i=B_i) = A_i \cdot B_i + A_i' \cdot B_i' = (A_i \oplus B_i)'$$

Por ejemplo, si  $A_3 = 1$  y  $B_3 = 1$ ;  $y_3$  será igual a  $y_3 = A_3 \cdot B_3 + A_3' \cdot B_3' = 1 \cdot 1 + 1 \cdot 1 = 1$  pero si  $A_3 = 1$  y  $B_3 = 0$ ;  $y_3 = A_3 \cdot B_3 + A_3' \cdot B_3' = 1 \cdot 0 + 0 \cdot 1 = 0$ . La comparación se realiza para el resto de los coeficientes  $A_i$  y  $B_i$ . El número  $A$  será igual a  $B$  si se cumple la condición  $y_i=1$  para todos los coeficientes, es decir una operación AND:

$$(A=B) = y_3 \cdot y_2 \cdot y_1 \cdot y_0$$

La variable binaria  $A=B$  es igual a 1 solamente si todos los pares de dígitos de los números son iguales.

Salidas  $A>B$  y  $A<B$

La comparación en este caso se comienza desde el bit más significativo. Los dígitos se comparan uno a uno y si estos son iguales se prueba con el siguiente par de bits menos significativos. La comparación continua hasta que se encuentra un par de dígitos desiguales. En la posición donde se encuentre un uno en  $A$  y un 0 en  $B$  se puede afirmar que  $A>B$ . Por el contrario, si  $A$  es igual a 0 y  $B$  igual a 1 entonces  $A<B$ . La función correspondiente a cada salida es:

$$(A>B) = A_3 \cdot B_3' + y_3 \cdot A_2 \cdot B_2' + y_3 \cdot y_2 \cdot A_1 \cdot B_1' + y_3 \cdot y_2 \cdot y_1 \cdot A_0 \cdot B_0'$$

$$(A<B) = A_3' \cdot B_3 + y_3 \cdot A_2' \cdot B_2 + y_3 \cdot y_2 \cdot A_1' \cdot B_1 + y_3 \cdot y_2 \cdot y_1 \cdot A_0' \cdot B_0$$

Ejemplo Comparar los números binarios  $A = A_3 \cdot A_2 \cdot A_1 \cdot A_0 = 1001$  y  $B = B_3 \cdot B_2 \cdot B_1 \cdot B_0 = 1011$ .

El valor de las variables  $y_i$ :

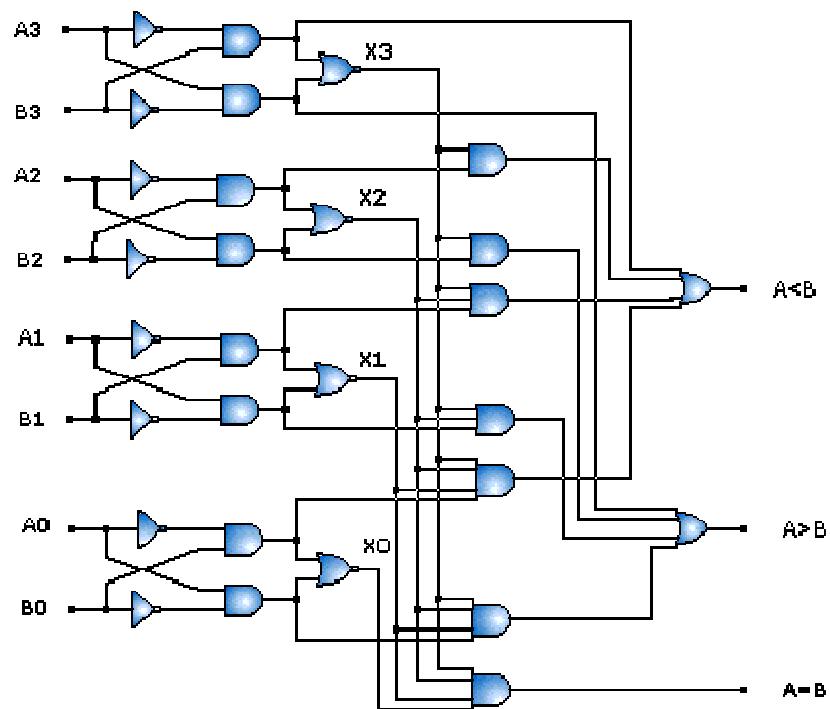
$$y_3(A_3=B_3) = (1) \cdot (1) + (0) \cdot (0) = 1 ; y_2(A_2=B_2) = (0) \cdot (0) + (1) \cdot (1) = 1 ; y_1(A_1=B_1) = (0) \cdot (1) + (1) \cdot (0) = 0 ; y_0(A_0=B_0) = (1) \cdot (1) + (1) \cdot (0) = 1.$$

Las ecuaciones son:

$$(A>B) = (1) \cdot (0) + (1) \cdot (0) \cdot (1) + (1) \cdot (1) \cdot (0) \cdot (0) + (1) \cdot (1) \cdot (0) \cdot (1) \cdot (0) = 0.$$

$$(A<B) = (0) \cdot (1) + (1) \cdot (1) \cdot (0) + (1) \cdot (1) \cdot (1) \cdot (1) + (1) \cdot (1) \cdot (0) \cdot (0) \cdot (1) = 1.$$

El diagrama del comparador de cuatro bits se muestra en la figura 97.



**Figura 97.** Comparador de magnitudes de cuatro bits

### 3.8. Implementación de Funciones Lógicas con Decodificadores

Teniendo en cuenta que los decodificadores son Circuitos integrados de Mediana Escala de Integración (MSI), se pueden implementar funciones lógicas con ellos, ya que en su interior existen entre 100 y 999 compuertas lógicas.

**Ejemplo:** A partir de la función de la Tabla 44, se explicará el procedimiento de diseño e implementación de Funciones Lógicas mediante decodificadores así (Ver figura 98):

A	B	C	Salida F	Numeración Salidas
0	0	0	0	S0
0	0	1	1	S1
0	1	0	0	S2
0	1	1	1	S3
1	0	0	0	S4
1	0	1	1	S5
1	1	0	0	S6
1	1	1	1	S7

**Tabla 44.** Tabla de Verdad Ejemplo Implementación de Funciones Lógicas con Decodificador.

#### Procedimiento:

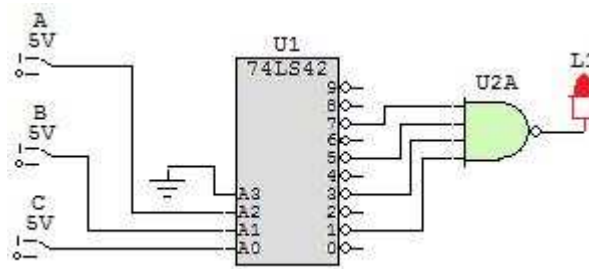
$$F = A'B'C + A'BC + AB'C + ABC$$

- Emplear un decodificador del mismo o mayor número de líneas de entrada que tenga la función a implementar. (Para este caso utilizaremos un decodificador 7442 que tiene 4 entradas y 10 Salidas) La entrada que no se necesite (La más significativa) se conecta a Tierra (GND).
- Buscar cada una de las salidas del decodificador que corresponde con la combinación de las variables de entrada que tienen un 1 en la salida. (Para este caso S1, S3, S5, S7).
- Para corregir la suma de términos de la salida F, se colocará una compuerta lógica que dependerá del codificador empleado teniendo en cuenta:

**COMPUERTA OR:** Para decodificadores con salidas activas en alto.

**COMPUERTA NAND:** Para decodificadores con salidas activas en bajo. (Esta es la que aplica para nuestro ejemplo, ya que, el CI7442 tiene sus salidas activas en Bajo).

- En caso que una o varias combinaciones de la tabla de verdad que tienen un 1 en su salida no correspondan con las salidas del decodificador, se añadirán las compuertas que representarán las combinaciones correspondientes.



**Figura 98.** Ejemplo Implementación de Funciones Lógicas con Decodificador

### 3.9. Implementación de Funciones Lógicas con Multiplexores

Así como los decodificadores, los Multiplexores son Circuitos integrados de Mediana Escala de Integración (MSI), se pueden implementar funciones lógicas con ellos, ya que en su interior existen entre 100 y 999 compuertas lógicas.

Para la implementación de funciones Lógicas con Multiplexores se tendrán en cuenta dos casos, según el número de entradas de selección de éste vs entradas de la función a implementar así:

1. Empleo de Multiplexores de igual número de entradas de selección que variables de entrada de la función a implementar
2. Empleo de Multiplexores con número inferior de entradas de selección que variables de entrada de la función a implementar

**Ejemplo:** A partir de la siguiente expresión algebraica de la función F1, se explicará el procedimiento de diseño e implementación de Funciones Lógicas mediante multiplexores así

$$F1 = A'B'C'D + A'B'CD + A'BCD' + A'BCD + A'BC'D + A'BC'D' + ABCD' + ABC'D' + AB'C'D$$

**Procedimiento Caso 1:** Empleo de Multiplexores de igual número de entradas de selección que variables de entrada de la función a implementar

A partir de la tabla de verdad de la Función F1 (Ver Tabla 45) se enumera en orden los pines de los datos de entrada del Multiplexor

Si el Valor de F1 correspondiente a las combinaciones de entrada está en uno (1) se conecta a Vcc (5V).

Si el Valor de F1 correspondiente a las combinaciones de entrada está en cero (0) se conecta a GND (Tierra).

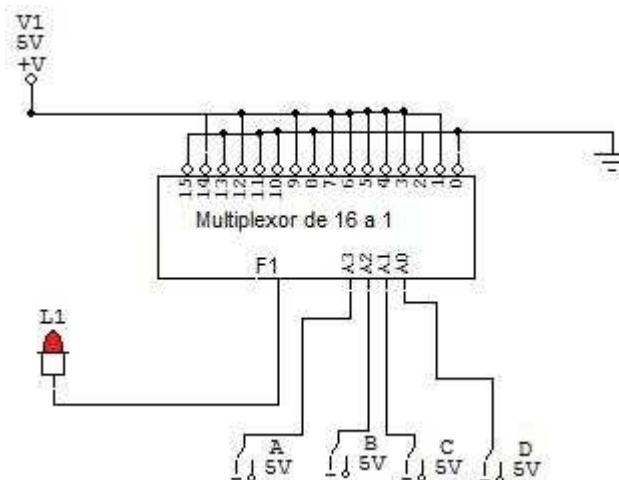
En la Figura 99 se puede observar el ejemplo del circuito, aquí se utiliza el CI 74LS150.

Bits de entrada = Canales de Selección del Multiplexor	Salida	Pines de Entrada de Datos del Multiplexor	Conexiones al Multiplexor
--	--------	---	---------------------------



A = S3	B = S2	C = S1	D = S0	F1	Datos	Conexión al Multiplexor
0	0	0	0	0	D0	GND
0	0	0	1	1	D1	VCC
0	0	1	0	0	D2	GND
0	0	1	1	1	D3	VCC
0	1	0	0	1	D4	VCC
0	1	0	1	1	D5	VCC
0	1	1	0	1	D6	VCC
0	1	1	1	1	D7	VCC
1	0	0	0	0	D8	GND
1	0	0	1	1	D9	VCC
1	0	1	0	0	D10	GND
1	0	1	1	0	D11	GND
1	1	0	0	1	D12	VCC
1	1	0	1	0	D13	GND
1	1	1	0	1	D14	VCC
1	1	1	1	0	D15	GND

**Tabla 45.** Empleo de Multiplexores de igual número de entradas de selección que variables de entrada de la función a implementar.



**Figura 99.** Ejemplo Implementación de Funciones Lógicas con Multiplexores de igual número de entradas de selección que variables de entrada de la función a implementar.

**Procedimiento Caso 2:** Empleo de Multiplexores con número inferior de entradas de selección que variables de entrada de la función a implementar

Se Dibuja una Tabla (de minterminos Ver Tabla 46) que represente las posiciones en orden colocando los valores de la función a implementar (o Tabla de Verdad), que para el caso del ejemplo 1 la expresión algebraica de F1 está dada por una suma de productos donde la función vale uno. Es importante aclarar que esta tabla no corresponde al Mapa de Karnaugh. El Circuito implementado se puede ver en la figura 100 ( con CI 74LS151).

En la Primera Columna y Primera Fila de la tabla 46 se pueden observar las variables de entrada de la Función, La más significativa (A) se deja aparte y le corresponderán los valores de 0 y 1 (Columna 1 filas 2 y 3), las demás variables menos significativas (BCD) corresponderán a las combinaciones (Fila 1 Columnas de la 2 hasta la 6)

En la tercera fila de la Tabla se colocaran en orden los Pines de Entrada del Multiplexor.

En la Cuarta Fila se indicará a qué conectar así:

Si los dos valores correspondientes al pin de entrada de datos (Columna del Dato) del Multiplexor (Filas 1 y 2) están en cero (0) se conecta a GND (Tierra).

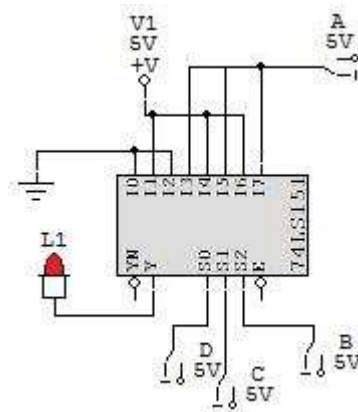
Si los dos valores correspondientes al pin de entrada de datos (Columna del Dato) del Multiplexor (Filas 1 y 2) están en uno (1) se conecta a GND (Tierra).

Si el Bit de la fila 2 está en uno (1) y el de la fila 2 está en cero (0) correspondientes al pin de entrada de datos (Columna del Dato), se conectará a A'.

Si el Bit de la fila 2 está en cero (0) y el de la fila 2 está en uno (1) correspondientes al pin de entrada de datos (Columna del Dato), se conectará a A.

BCD A	000	001	010	011	100	101	110	111
0	0	1	0	1	1	1	1	1
1	0	1	0	0	1	0	1	0
Pines de Entrada del Mux	D0	D1	D2	D3	D4	D5	D6	D7
Conexión para obtener la función	GND	Vcc	GND	A'	Vcc	A'	Vcc	A'

**Tabla 46.** Tabla de Verdad Ejemplo Implementación de Funciones Lógicas con Decodificador.



**Figura 100.** Ejemplo Implementación de Funciones Lógicas Empleo de Multiplexores con número inferior de entradas de selección que variables de entrada de la función a implementar

#### 4. Lógica Secuencial

En la lógica secuencial a diferencia de la lógica combinatoria se hace uso de un elemento básico llamado *flip-flop*. El *FLIP-FLOP* es un elemento de memoria que almacena un *bit* de información. Algunos textos usan este nombre para referirse a los *cerrojos*, pero en la mayoría de las publicaciones se hace la diferencia entre *FLIP-FLOP* y *latch*. Este último término es el que traducimos como *cerrojo*.

Los circuitos lógicos secuenciales tienen la capacidad de memorizar información, en consecuencia, los valores de las salidas, en un determinado momento, no dependen exclusivamente de los valores de las entradas en ese instante, sino que dependen también de los que estuvieron presentes con anterioridad.

Los circuitos lógicos secuenciales se dividen básicamente en dos grupos: Los circuitos asincrónicos y los circuitos sincrónicos. Los primeros pueden cambiar los estados de sus salidas como resultado del cambio de los estados de las entradas, mientras que los circuitos sincrónicos pueden cambiar el estado de sus salidas en instantes de tiempo discretos bajo el control de una señal de reloj.

Existen tres circuitos clasificados según la forma en que retienen o memorizan el estado que adoptan sus salidas, estos son:

**Circuitos Biestables o FLIP-FLOP (FF):** Son aquellos que cambian de estado cada vez que reciben una señal de entrada (ya sea nivel bajo o alto), es decir retienen el dato de salida aunque desaparezca el de entrada. **Conclusión: Poseen dos estados estables**



**Circuitos Monoestables:** Estos circuitos cambian de estado sólo si se mantiene la señal de entrada (nivel alto o bajo), si ésta se quita, la salida regresa a su estado anterior. **Conclusión: Poseen un sólo estado estable y otro metaestables.**

**Circuitos Astables o Aestables:** Son circuitos gobernados por una red de tiempo R-C (Resistencia-Capacitor) y un circuito de realimentación, a diferencia de los anteriores se puede decir que no poseen un estado estable sino dos metaestables.



##### 4.1. Oscilador Simétrico con compuertas NOT:

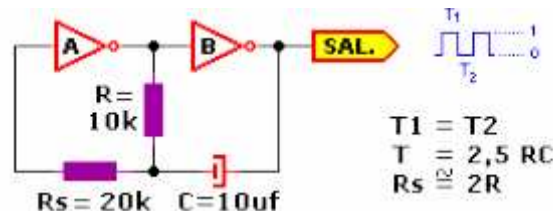
Supongamos que en determinado momento la salida del inversor B está a nivel "1", entonces su entrada está a "0", y la entrada del inversor "A" a nivel "1". En esas condiciones C se carga a través de R, y los inversores permanecen en ese estado.

Cuando el condensador alcanza su carga máxima, se produce la conmutación del inversor "A". Su entrada pasa a "0", su salida a "1" y la salida del inversor "B" a "0", se invierte la polaridad del capacitor y este se descarga, mientras tanto los inversores permanecen sin cambio, una vez descargado, la entrada del inversor "A" pasa nuevamente a "1", y comienza un nuevo ciclo.

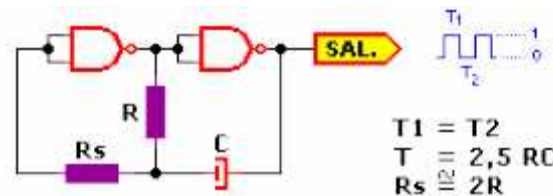
Este oscilador es simétrico ya que el tiempo que dura el nivel alto es igual al que permanece en nivel bajo, este tiempo está dado por  $T = 2,5 RC$

T expresado en segundos, R en Ohms, C en Faradios

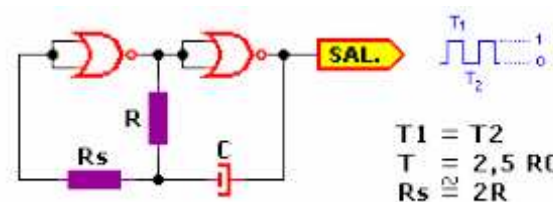
Ahora bien, si recordamos las leyes de De Morgan, uniendo las entradas de compuertas NAND o compuertas NOR se obtiene la misma función que los inversores o compuertas NOT.



**Figura 101.** Oscilador Simétrico con compuertas NOT



**Figura 102.** Oscilador Simétrico con compuertas NAND

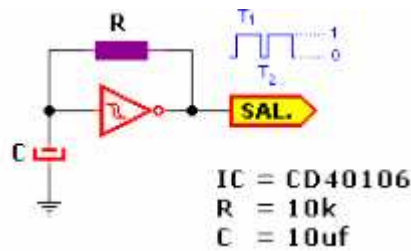


**Figura 103.** Oscilador Simétrico con compuertas NOR

## 4.2. Disparadores Schmitt Trigger

Las compuertas SCHMITT TRIGGER o disparadores de Schmitt, son iguales a las compuertas vistas hasta ahora, pero tienen la ventaja de tener umbrales de conmutación muy definidos llamados  $V_{T+}$  y  $V_{T-}$ , esto hace que puedan reconocer señales que en las compuertas lógicas comunes serían una indeterminación de su

estado y llevarlas a estados lógicos definidos, mucho más definidos que las compuertas comunes que tienen un solo umbral de conmutación.



**Figura 104.** Disparador SCHMITT TRIGGER

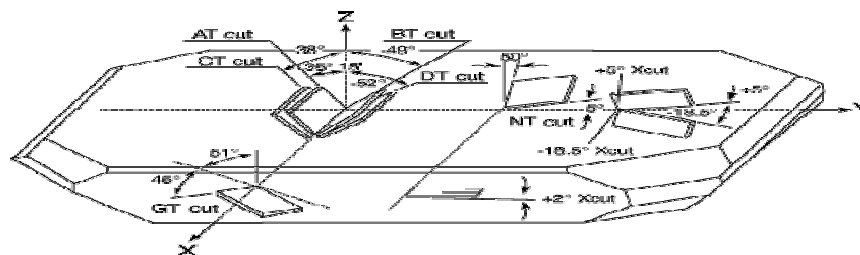
Si la salida está en un nivel lógico 1, C comienza a cargarse a través de R, a medida que la tensión crece en la entrada de la compuerta, esta alcanza el nivel  $V_{T+}$  y produce la conmutación de la compuerta llevando la salida a nivel 0 y el capacitor comienza su descarga. Cuando el potencial a la entrada de la compuerta disminuye por debajo del umbral de  $V_{T-}$ , se produce nuevamente la conmutación pasando la salida a nivel 1, y se reinicia el ciclo.

Integrado	Frecuencia	Valor de R
7414	$0.8/RC$	$R \leq 500W$
74LS14	$0.8/RC$	$R \leq 2W$
74HC14	$1.2/RC$	$R \leq 10MW$

**Tabla 47.** Frecuencia de oscilación según R y C

### 4.3. Oscilador de Cristal

El cristal de cuarzo es utilizado como componente de control de la frecuencia de circuitos osciladores convirtiendo las vibraciones mecánicas en voltajes eléctricos a una frecuencia específica. Esto ocurre debido al efecto "piezoeléctrico". La piezo-electricidad es electricidad creada por una presión mecánica. En un material piezoeléctrico, al aplicar una presión mecánica sobre un eje, dará como consecuencia la creación de una carga eléctrica a lo largo de un eje ubicado en un ángulo recto respecto al de la aplicación de la presión mecánica. En algunos materiales, se encuentra que aplicando un campo eléctrico según un eje, produce una deformación mecánica según otro eje ubicado a un ángulo recto respecto al primero. Por las propiedades mecánicas, eléctricas, y químicas, el cuarzo es el material más apropiado para fabricar dispositivos con frecuencia bien controlada.



**Figura 105.** Ubicación de elementos específicos dentro de una piedra de cuarzo

**Frecuencia Fundamental vs. Frecuencia de Sobretono:** Esto es de importancia cuando se especifica un cristal. Cuando se incrementa la frecuencia solicitada, el espesor del cuerpo del cristal disminuye y por supuesto existe un límite en el proceso de fabricación. Alrededor de 30MHz, el espesor de la placa del cristal comienza a ser muy delgada. Debido a que el corte "AT" (ver figura 105) resonará a números enteros impares múltiplos de la frecuencia fundamental, es necesario especificar el orden del sobretono deseado para cristales de altas frecuencias.

**Potencia de trabajo (Drive Level):** Es la potencia disipada por el cristal. Está normalmente especificada en micro o mili vatios, siendo un valor típico 100 micro vatios.

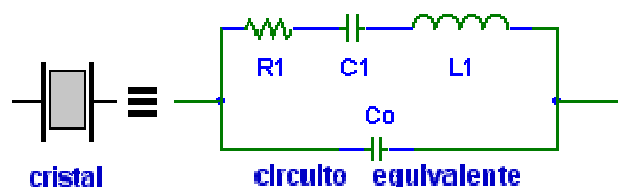
**Tolerancia en la frecuencia:** La tolerancia en la frecuencia se refiere a la máxima desviación permitida y se expresa en partes por millón (PPM) para una temperatura especificada, usualmente 25°C.

**Estabilidad de la frecuencia:** La estabilidad de la frecuencia se refiere a la máxima desviación en PPM, en un determinado rango de temperatura. La desviación está tomada con referencia a la frecuencia medida a 25°C.

**Envejecimiento:** El envejecimiento se refiere a los cambios acumulativos en la frecuencia del cristal con el transcurrir del tiempo. Los factores que intervienen son: exceso en la potencia disipada, efectos térmicos, fatiga en los alambres de armado y pérdidas en la elasticidad del cristal. El diseño de circuitos considerando bajas temperaturas ambientales y mínimas potencias en el cristal reducirán el envejecimiento.

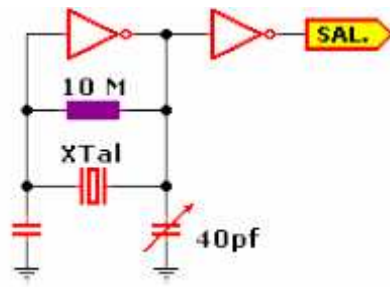
**Circuito Eléctrico Equivalente:** El circuito eléctrico equivalente que se muestra a continuación es un esquema del cristal de cuarzo trabajando a una determinada frecuencia de resonancia. El condensador  $C_o$  en paralelo, representa en total la capacidad entre los electrodos del cristal más la capacidad de la carcasa y sus terminales.  $R_1$ ,  $C_1$  y  $L_1$  conforman la rama principal del cristal y se conocen como componentes o parámetros motional donde:

- $L_1$  representa la masa vibrante del cristal,
- $C_1$  representa la elasticidad del cuarzo y
- $R_1$  representa las pérdidas que ocurren dentro del cristal.

**Figura 106.** Circuito Eléctrico Equivalente de un cristal de cuarzo

Por ejemplo, un oscilador implementado con dos inversores y un Cristal de cuarzo, el trimer de 40pf se incluye para un ajuste fino de la frecuencia de oscilación, mientras el

circuito oscilante en si funciona con un solo inversor, se incluye otro para actuar como etapa separadora.



**Figura 107.** Oscilador a Cristal

#### 4.4. Osciladores Controlados

Se trata simplemente de controlar el momento en que estos deben oscilar, tenemos dos opciones, que sean controlados por un nivel alto o por un nivel bajo.

Se tiene en cuenta que los osciladores vistos hasta el momento solo pueden oscilar cambiando el estado de sus entradas en forma alternada, lo que haremos será forzar ese estado a un estado permanente, como dije anteriormente ya sea a 1 o 0.

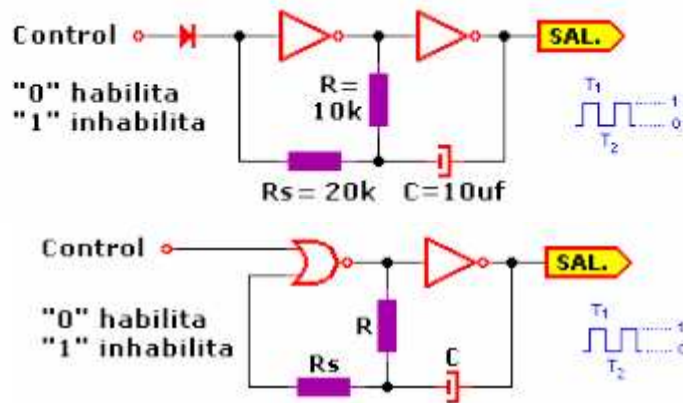


Figura 106. Osciladores Controlados a Nivel 0

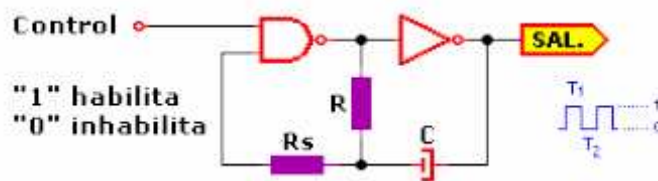


Figura 107. Osciladores Controlados a Nivel 1

#### 4.5. Circuito Integrado 555

Este Circuito Integrado (C.I.) es para los experimentadores y aficionados un dispositivo barato con el cual pueden hacer muchos proyectos. Es un temporizador es tan versátil que se puede, incluso utilizar para modular una señal en frecuencia modulada (F.M.)

Está constituido por una combinación de comparadores lineales, Flip-Flops (básculas digitales), transistor de descarga y excitador de salida.

Es muy popular para hacer osciladores que sirven como reloj (base de tiempo) para el resto del circuito. A continuación se explicará la configuración de sus pines:



Figura 108. Representaciones del CI 555



**Pin 1 - Tierra o masa**

**Pin 2 - Disparo:** Es en esta patilla, donde se establece el inicio del tiempo de retardo, si el 555 es configurado como monostable. Este proceso de disparo ocurre cuando este pin va por debajo del nivel de 1/3 del voltaje de alimentación. Este pulso debe ser de corta duración, pues si se mantiene bajo por mucho tiempo la salida se quedará en alto hasta que la entrada de disparo pase a alto otra vez.

**Pin 3 - Salida:** Aquí veremos el resultado de la operación del temporizador, ya sea que esté conectado como monostable, astable u otro. Cuando la salida es alta, el voltaje será el voltaje de aplicación (Vcc) menos 1.7 Voltios. Esta salida se puede obligar a estar en casi 0 voltios con la ayuda de la patilla reset ( Pin 4)

**Pin 4 - Reset:** Si se pone a un nivel por debajo de 0.7 Voltios, pone la patilla de salida 3 a nivel bajo. Si por algún motivo esta patilla no se utiliza hay que conectarla a Vcc para evitar que el 555 se "reinicie"

**Pin 5 - Control de voltaje:** Cuando el temporizador se utiliza en el modo de controlador de voltaje, el voltaje en esta patilla puede variar casi desde Vcc (en la práctica como Vcc-1 voltio) hasta casi 0 V (aprox. 2 Voltios). Así es posible modificar los tiempos en que la patilla 3 está en alto o en bajo independiente del diseño (establecido por las resistencias y condensadores conectados externamente al 555). El voltaje aplicado a la patilla 5 puede variar entre un 45 y un 90 % de Vcc en la configuración monoestable. Cuando se utiliza la configuración astable, el voltaje puede variar desde 1.7 voltios hasta Vcc. Modificando el voltaje en esta patilla en la configuración astable causará la frecuencia original del astable sea modulada en frecuencia (FM). Si esta patilla no se utiliza, se recomienda ponerle un condensador de 0.01uF para evitar las interferencias.

**Pin 6 - Umbral:** Es una entrada a un comparador interno que tiene el 555 y se utiliza para poner la salida (Pin 3) a nivel bajo.

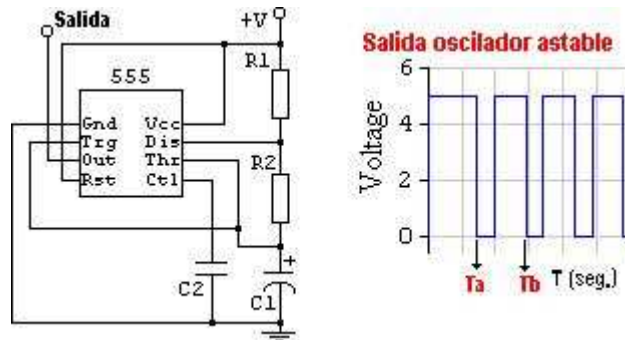
**Pin 7 - Descarga:** Utilizado para descargar con efectividad el condensador externo utilizado por el temporizador para su funcionamiento.

**Pin 8 - V+:** También llamado Vcc, es el pin donde se conecta el voltaje de alimentación que va de 4.5 voltios hasta 16 voltios (máximo). Hay versiones militares de este integrado que llegan hasta 18 Voltios.

El CI 555 se puede conectar para que funcione de diferentes maneras, entre los más importantes están: como multivibrador astable y como multivibrador monoestable.

**4.6. CI 555 como Multivibrador Astable:** Este tipo de funcionamiento se caracteriza por una salida con forma de onda cuadrada (o rectangular) continua de ancho predefinido por el diseñador del circuito. El esquema de conexión es el que se muestra. La señal de salida tiene un nivel alto por un tiempo T1 y en un nivel bajo un tiempo T2. Los tiempos de duración dependen de los valores de R1 y R2.

$$T1 = 0.693(R1+R2)C1 \text{ y } T2 = 0.693 \times R2 \times C1 \text{ (en segundos)}$$



**Figura 109.** CI 555 como Multivibrador Astable

La frecuencia con que la señal de salida oscila está dada por la fórmula:

$$f = 1 / [0.693 \times C1 \times (R1 + 2 \times R2)]$$

$$\text{y el período es simplemente } T = 1 / f$$

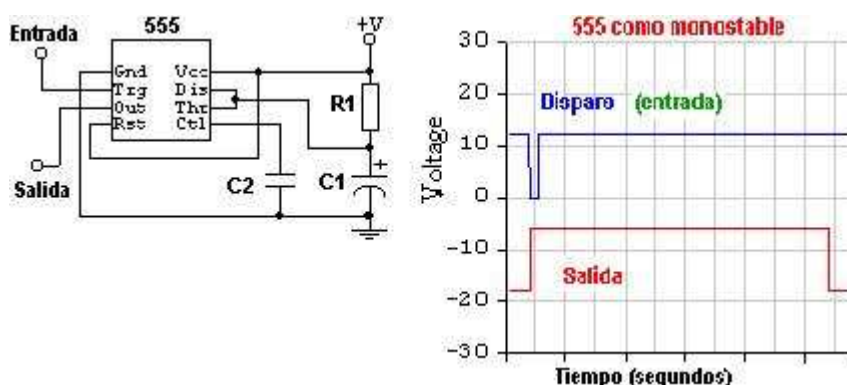
Hay que recordar que el período es el tiempo que dura la señal hasta que ésta se vuelve a repetir ( $T_b - T_a$ ), ver figura 109.

**4.7. CI 555 como Multivibrador Monoestable:** En este caso el circuito entrega a su salida un solo pulso de un ancho establecido por el diseñador (tiempo de duración) Ver figura 110.

El esquema de conexión es el que se muestra. La Fórmula para calcular el tiempo de duración (tiempo que la salida está en nivel alto) es:

$$T = 1.1 \times R1 \times C1 \text{ (en segundos).}$$

Observar que es necesario que la señal de disparo, sea de nivel bajo y de muy corta duración en el PIN 2 del C.I. para iniciar la señal de salida.



**Figura 110.** CI 555 como Multivibrador Monoestable

#### 4.8. Circuitos Monoestables:

**Monoestable sencillo con un inversor:** Considere inicialmente la entrada del inversor en nivel bajo a través de R y C, entonces su salida estará a nivel alto, ahora bien, un 1 lógico de poca duración en la entrada, hace que se cargue el capacitor y

conmute el inversor entregando un 0 lógico en su salida, y este permanecerá en ese estado hasta que la descarga del capacitor alcance el umbral de histéresis de la compuerta y entonces conmutará y regresará a su estado inicial (Ver figura 111)

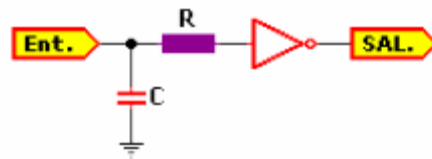


Figura 111. Monoestable sencillo con inversor

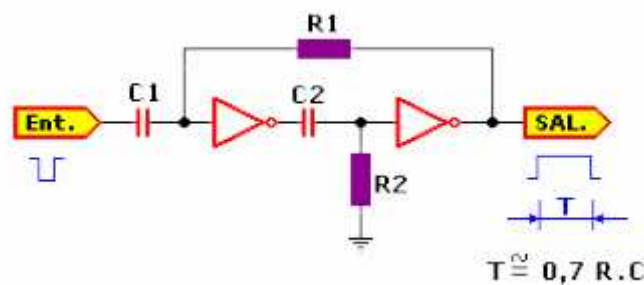


Figura 112. Monoestable con dos inversores

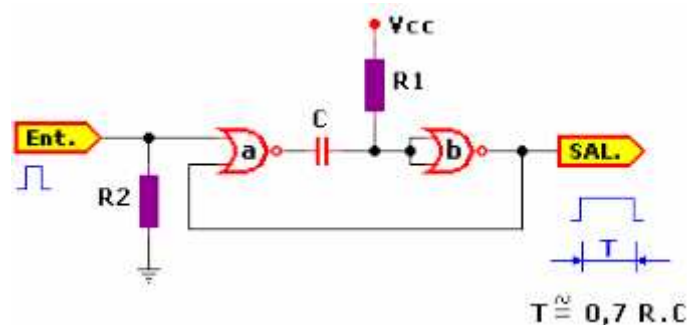


Figura 113. Monoestable con dos compuertas NOR

#### 4.9. Circuitos Biestables (FLIP-FLOPs):

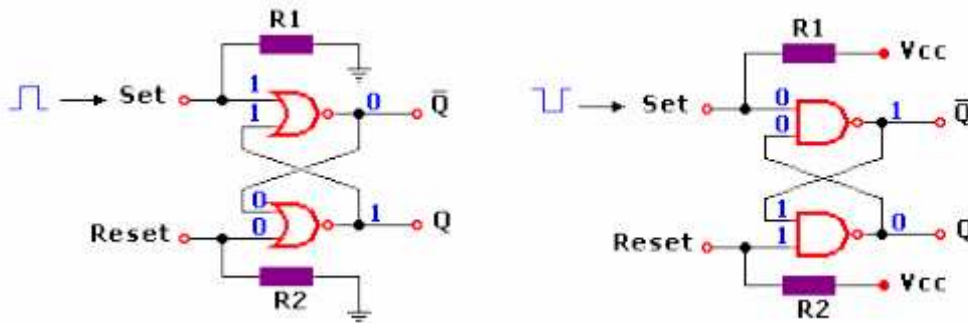
Los circuitos biestables son muy conocidos y empleados como elementos de memoria, ya que son capaces de almacenar un bit de información. En general, son conocidos como FLIP-FLOP y poseen dos estados estables, uno a nivel alto (1 lógico) y otro a nivel bajo (cero lógico).

**Observación:** es posible que al presionar el pulsador se produzcan rebotes eléctricos, es como haberlo presionado varias veces, y sí... los resultados serán totalmente inesperados, así que si se utilizan los cables para probar estos circuitos no nos servirán de mucho, es conveniente utilizar un pulso de reloj para realizar estas pruebas, un circuito astable o monoestable, que llamaremos pulso de reloj o Clock o CK.

Por lo general un FLIP-FLOP dispone de dos señales de salida, una con el mismo valor de la entrada y otra con la negación del mismo o sea su complemento.

Existen varios tipos de FLIP-FLOPs y variaciones de estos que permiten realizar funciones específicas, dependiendo de la aplicación. A continuación veremos algunos de ellos.

**4.9.1. FLIP-FLOP Básico R-S (Reset-Set):** Se puede construir uno fácilmente utilizando dos compuertas NAND o NOR conectadas de tal forma de realimentar la entrada de una con la salida de la otra, quedando libre una entrada de cada compuerta, las cuales serán utilizadas para control Set y Reset.



**Figura 114.** FLIP-FLOP Básico R-S con compuertas NOR y NAND

Las resistencias R1 y R2 utilizadas en ambos casos son de 10k y empleadas solamente para evitar estados indeterminados, observa el circuito con compuertas NOR... Un nivel alto aplicado en Set, hace que la salida negada  $Q'$  sea 0 debido a la tabla de verdad de la compuerta NOR, al realimentar la entrada de la segunda compuerta y estando la otra a masa, la salida normal Q será 1. Ahora bien, esta señal realimenta la primer compuerta, por lo tanto no importan los rebotes, y el FF se mantendrá en este estado hasta que le des un pulso positivo a la entrada Reset

**Conclusión:** El Biestable posee dos entradas Set y Reset que trabajan con un mismo nivel de señal, provee dos salidas, una salida normal Q que refleja la señal de entrada Set y otra  $Q'$  que es el complemento de la anterior. Si comparas los dos FLIP-FLOP representados en la figura 111, verás que sólo difieren en los niveles de señal que se utilizan, debido a la tabla de verdad que le corresponde a cada tipo de compuerta. Este suele presentar un estado indeterminado cuando sus dos entradas R y S se encuentran en estado alto (ver tablas 47 y 48).

$S_i$	$R_i$	$Q_{i+1}$
0	0	$Q_i$
0	1	0
1	0	1
1	1	-

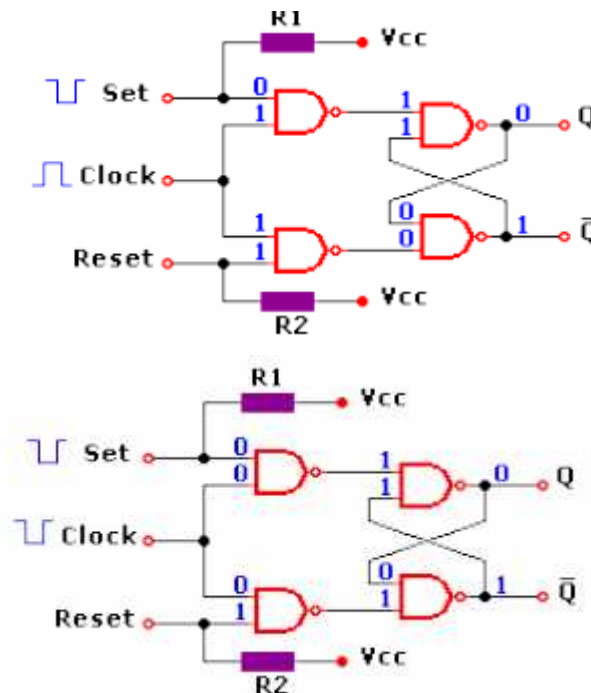
**Tabla 48.** Estados lógicos del FLIP-FLOP R-S con compuertas NAND

$S_i$	$R_i$	$Q_{i+1}$
0	0	-

0	1	0
1	0	1
1	1	$Q_i$

**Tabla 49.** Estados lógicos del FLIP-FLOP R-S con compuertas NOR

**4.9.2. FLIP FLOP RS - Controlado por un pulso de reloj:** En este caso voy a utilizar el ejemplo de las compuertas NAND, pero le agregaremos dos compuertas más, y uniremos la entrada de cada una a una señal de Reloj



**Figura 115.** FLIP-FLOP R-S Controlado por un pulso de reloj

Necesitamos un generador de pulsos (Astable) para conectarlo en la entrada Clock, una vez lo tenemos pasamos a interpretar el circuito...

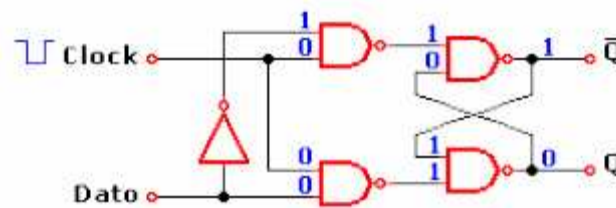
Sorpresa, el FF se mantiene sin cambios en Q y Q'. Fíjate que ahora no importa el estado de Set y Reset, esto se debe a su tabla de verdad (basta que una de sus entradas sea 0 para que su salida sea 1) por lo tanto Set y Reset quedan inhabilitadas. Es decir que se leerán los niveles de Set y Reset sólo cuando la entrada Clock sea 1.

**NOTA 1:** El primer circuito que vimos (FLIP-FLOP simple) es llamado FLIP-FLOP Asíncrono ya que puede cambiar el estados de sus salidas en cualquier momento, y sólo depende de las entradas Set y Reset.

**NOTA 2:** El segundo circuito es controlado por una entrada Clock y es llamado FLIP-FLOP Síncrono ya que el cambio de estado de sus salidas esta sincronizado por un pulso de reloj que realiza la lectura de las entradas en un determinado instante.

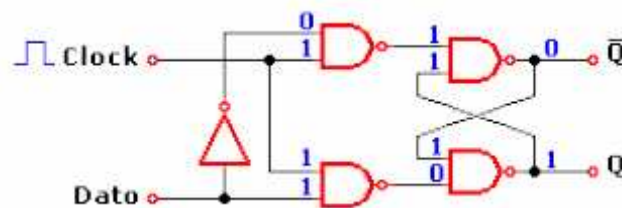
Si pones un 0 en Set y la entrada Clock está a 1 ocurrirá todo lo que se describe en el esquema anterior, veamos figura 116 que ocurre cuando Clock pasa a 0...

**4.9.3. FLIP-FLOP D:** En este circuito no existe la posibilidad de que las dos entradas estén a nivel alto ya que posee un inversor entre la una y la otra de tal modo que  $R = \sim S$ , observa la figura 116, aquí se supone la entrada Dato a nivel 0...



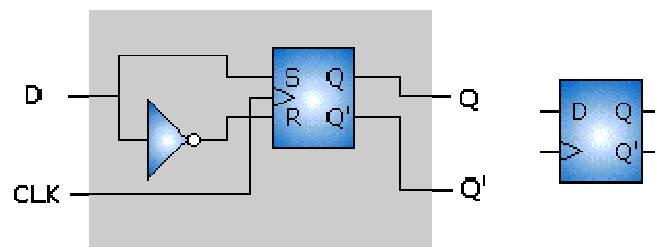
**Figura 116.** FLIP-FLOP D con entrada Dato a nivel 0

Veamos que ocurre cuando la entrada Dato, pasa a 1 y CK cambia de estado pasando también a 1, según como se van transmitiendo los datos por las compuertas resulta  $Q = 1$  y  $Q' = 0$ .



**Figura 117.** FLIP-FLOP D con entrada Dato a nivel 1

Para que el FLIP-FLOP retorne a su estado inicial, la entrada Dato D deberá pasar a 0 y sólo se transferirá a la salida si CK es 1. Nuevamente se repite el caso que para leer el datos debe ser  $CK = 1$ . En forma general se representa el FLIP-FLOPD con el siguiente símbolo



**Figura 118.** Símbolo FLIP-FLOP D

D	CLK	$Q_{i+1}$
0	↑	0
1	↑	1

**Tabla 50.** Estados del FLIP-FLOP D

La forma de operación de este *FLIP-FLOP* es muy sencilla:

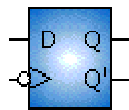
- Cuando  $D=0$  y se presenta un cambio de 0 a 1 lógico en la entrada de reloj del *FLIP-FLOP* la salida  $Q=0$ .

- Cuando  $D=1$  y se presenta un cambio de 0 a 1 lógico en la entrada de reloj del *FLIP-FLOP* la salida  $Q=1$ .

En otras palabras, el dato en  $D$  se transfiere y memoriza en  $Q$  cada vez que se presenta una transición de 0 a 1 lógico en la señal de reloj ( $CLK$ ); esta condición se conoce con el nombre de transición por flanco positivo.

La condición complementaria a la anterior es cuando la transición es de 1 a 0 lógico, en este caso se dice que la transición se da por flanco negativo.

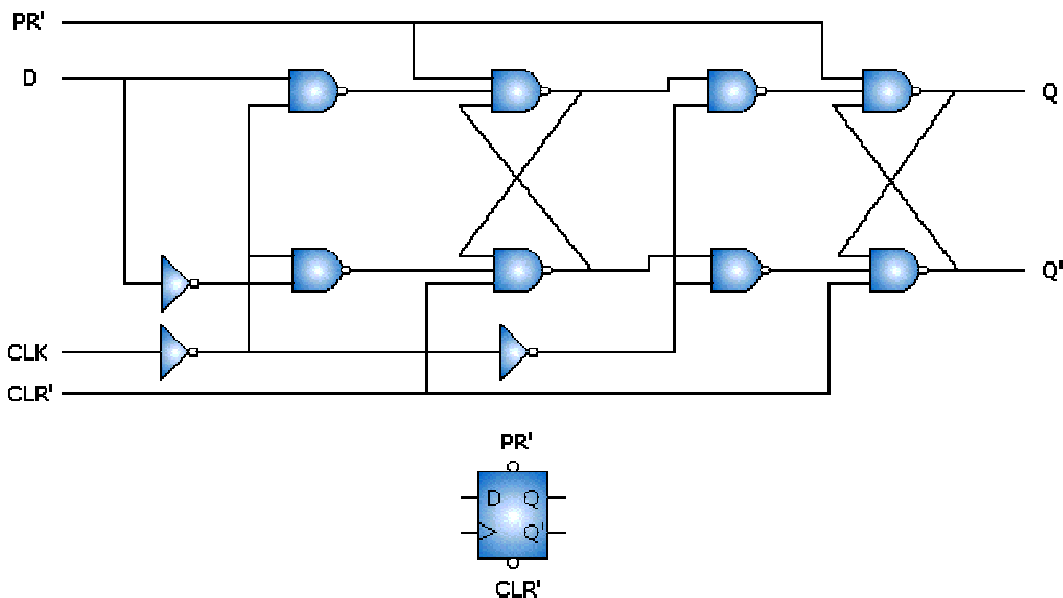
Este *FLIP-FLOP* se puede utilizar para que la transición se de por flanco negativo, simplemente basta con poner a la entrada del reloj ( $CLK$ ) un inversor como en la figura 119.



**Figura 119.** FLIP-FLOP D con inversor en la entrada de reloj

#### 4.9.4. FLIP-FLOP D PRESET-CLEAR

Este *FLIP-FLOP* es similar al *FLIP-FLOP D*, excepto que este tiene dos entradas asincrónicas activadas en bajo llamadas *Preset* y *Clear*. Estas entradas como su nombre lo indican sirven respectivamente para poner en 1 y 0 la salida  $Q$  del *FLIP-FLOP* independientemente de la señal de reloj. La configuración de este *FLIP-FLOP* y su representación abreviada se describen en la figura 120

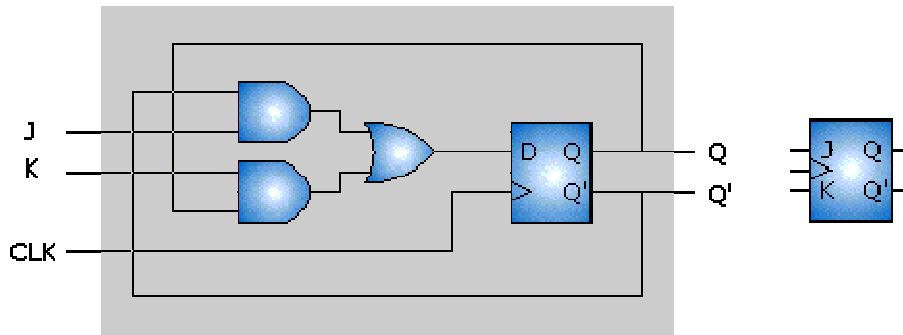


**Figura 120.** FLIP-FLOP D Preset-Clear

La gran parte de los Circuitos Integrados que contienen *FLIP-FLOPs* vienen con entradas asincrónicas de inicialización y borrado (*Preset* y *Clear*), comúnmente representados con las abreviaturas *PRE* y *CLR*.

#### 4.9.5. FLIP-FLOPJ-K

Este *FLIP-FLOP* es una versión modificada del *FLIP-FLOP D*, y su aplicación es muy difundida en el Análisis y Diseño de Circuitos Secuenciales. El funcionamiento de este dispositivo es similar al *FLIP-FLOP S-R*, excepto que en este no se presentan indeterminaciones cuando sus dos entradas se encuentran en 1 lógico, si no que el *FLIP-FLOP* entra en un modo de funcionamiento llamado *modo complemento*, en el cual, la salida  $Q$  cambia a su estado complementario después de cada pulso de reloj. La configuración de este *FLIP-FLOP* y su representación abreviada se muestran en la figura 121 y en la tabla 51 se indican los estados de entrada y salida de este *FLIP-FLOP*.



**Figura 121.** Representación del FLIP-FLOP J-K

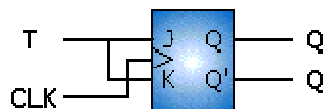
Note que las entradas  $J$  y  $K$  controlan el estado de este *FLIP-FLOP* de la misma manera que en el *FLIP-FLOP D*. Cuando las entradas son  $J=1$  y  $K=1$  no generan un estado indeterminado a la salida, sino que hace que la salida del *FLIP-FLOP* cambie a su estado complementario.

J	K	CLK	$Q_{i+1}$
0	0	↑	$Q_i$
1	0	↑	1
0	1	↑	0
1	1	↑	$Q_i'$

**Tabla 51.** Estados del FLIP-FLOP J-K

#### 4.9.6. FLIP-FLOP T (Toggle)

Este *FLIP-FLOP* recibe su nombre por la función que realiza (*Toggle*) cambiando el estado de la salida por su complemento. Es una modificación del *FLIP-FLOP J-K* limitándolo a cumplir exclusivamente esta función, la cual se logra uniendo las terminales  $J$  y  $K$  como se muestra en la figura 122.



**Figura 122.** FLIP-FLOP T

La tabla de verdad de este *FLIP-FLOP* se limita a las líneas 1 y 4 del *FLIP-FLOP J-K*.



T	CLK	$Q_{i+1}$
0	↑	$Q_i$
1	↑	$Q_i'$

Tabla 52. Estados del FLIP-FLOP T

## 5. Contadores y Registros

Son circuitos digitales lógicos secuenciales de salida binaria o cuenta binaria, característica de temporización y de memoria, por lo cual están constituidos a base de flip-flops.

**Características Importantes:** 1. Un número máximo de cuentas (módulo del contador), 2. Cuenta ascendente o descendente, 3. Operación síncrona o asíncrona, 4. Autónomos o de autodetención.

**Utilidad** Se utilizan para contar eventos. **Ejemplos:** 1. Número de pulsos de reloj, 2. Medir frecuencias, 3. Se utilizan como divisores de frecuencia y para almacenar datos (en un reloj digital), 4. Se utilizan para direccionamiento secuencial y algunos circuitos aritméticos.

**5.1. Contadores de Propagación:** Los contadores digitales o binarios en esencia son un grupo de *FLIP-FLOPs* dispuestos de tal manera que sus salidas proporcionan una secuencia determinada como respuesta a los acontecimientos que ocurren a la entrada del reloj. Estos acontecimientos pueden ser por lo general pulsos de reloj (sincrónicos) o acontecimientos aleatorios (asincrónicos) alimentados como entradas por la terminal de reloj de los *FLIP-FLOPs*. Los contadores de propagación se basan en este último principio para generar secuencias binarias que cambian como respuesta a eventos.

Para conformar un contador de  $n$  bits solo basta tener  $n$  *FLIP-FLOPs*, uno para cada *BIT* de información. A continuación se dará una descripción sobre la estructura y funcionamiento de los contadores de propagación más comunes en lógica secuencial.

**5.2. Contador de propagación ascendente:** El *FLIP-FLOP T*, tiene especial aplicación en los contadores, debido a la habilidad que tienen para cambiar a su estado complementario, después de un evento de reloj.

Número de pulsos	$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
...	...	...	...	...
15	1	1	1	1
16	0	0	0	0
17	0	0	0	1
...	...	...	...	...

**Tabla 53.** Estados contador ascendente

Observe la forma en que opera este circuito. Los pulsos de reloj se aplican únicamente al *FLIP-FLOP* A, así que la salida de este *FLIP-FLOP* se complementará cada vez que haya una transición negativa en la entrada de reloj.

La salida del *FLIP-FLOP* A se aplica directamente a la entrada de reloj del *FLIP-FLOP* B, de tal forma que la salida de este *FLIP-FLOP* se complementa cada vez que su entrada de reloj pasa de 1 a 0 lógico. De forma similar se comportan los *FLIP-FLOPs* C y D cambiando su estado cada vez que reciben una transición negativa en sus respectivas entradas de reloj.

Las salidas de los *FLIP-FLOPs* D, C, B y A representan un número binario de 4 bits, siendo D el bit más significativo y al menos significativo.

Este contador cuenta en forma ascendente desde 0000 hasta 1111, es decir que tiene 16 estados diferentes ( $2^4=16$ ). En electrónica digital, existe una notación que define el número de estados de un contador, designada por la sigla MOD más el número de estados, por esta razón se dice que es un contador MOD16. Este tipo de contadores actúa como divisores de frecuencia. Si se hace un análisis sobre la frecuencia de las señales de salida de los *FLIP-FLOPs* se puede observar que la señal  $Q_3$  tiene una frecuencia dada por la siguiente expresión:

$$f_{Q3} = \frac{f_{clk}}{16}$$

Donde  $f_{CLK}$  corresponde a la frecuencia de la señal del reloj. De igual forma las frecuencias de las salidas de los demás *FLIP-FLOPs* estarían dadas por las siguientes expresiones:

$$f_{Q2} = \frac{f_{clk}}{8}$$

$$f_{Q1} = \frac{f_{clk}}{4}$$

$$f_{Q0} = \frac{f_{clk}}{2}$$

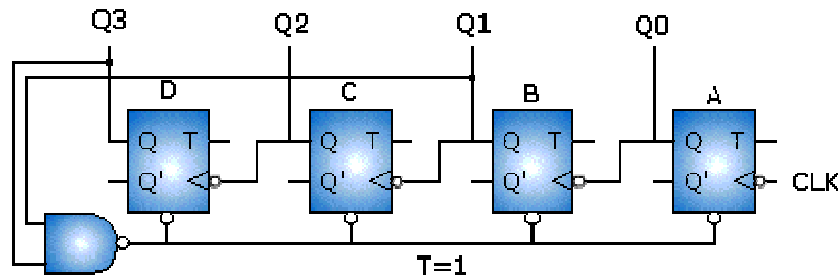
Se plantea como ejercicio dibujar la señal de reloj y las señales de salida de los *FLIP-FLOPs* para confirmar estos resultados.

Este contador se puede modificar para que opere a cualquier número MOD entre 1 y 16. De forma general un contador de  $n$  bits se puede modificar para cualquier número  $MOD2^n$ , y para lograrlo es necesario utilizar la entrada asincrónica de borrado CLR de los *FLIP-FLOPs*, como veremos a continuación.

### 5.3. Contadores con números MOD < $2^n$

Los contadores básicos pueden ser modificados para producir números MOD <  $2^n$ , permitiendo que el contador omita estados que normalmente hacen parte de la secuencia de conteo. La forma más usual para lograr esto se puede ver en la Figura

123, la cual corresponde a un contador de 4 bits *MOD10*. Este contador es conocido también como contador décadas.



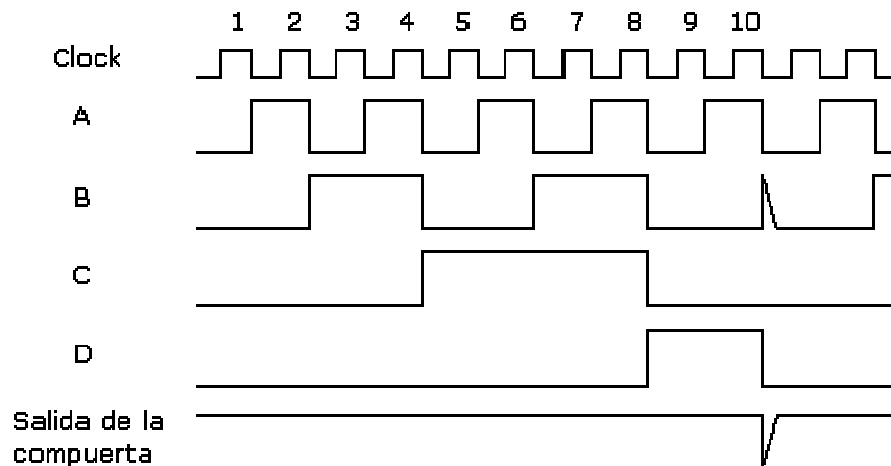
**Figura 123.** Contador décadas (MOD10)

Asumiendo que la compuerta *NAND* no estuviera presente, el contador sería *MOD16*, sin embargo la presencia de esta compuerta altera el funcionamiento normal cuando las salidas  $Q_3$  y  $Q_1$  que van a la compuerta son 1. Esta condición ocurrirá cuando el contador pase del estado 1001 (9) al 1010 (10), haciendo que las entradas asíncronas *CLR* de los *FLIP-FLOPs* sean 0 y por tanto el contador pase al estado 0000. En la Tabla 54, se resumen los estados de este contador.

En el momento que el contador llega al estado 1001 y ocurre una nueva transición en la entrada de reloj (*CLK*), se presenta el estado 1010 (10) de forma temporal, y su duración depende del tiempo de propagación de la compuerta *NAND*. En la Figura 123 se observa el estado temporal entre los estados 1001 y 0000.

Número de pulsos	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
...	...	...	...	...
9	1	0	0	1
10	0	0	0	0
11	0	0	0	1
...	...	...	...	...

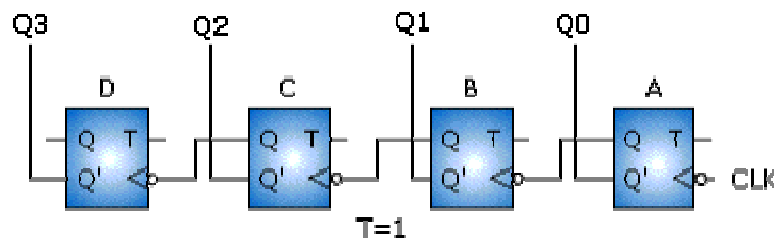
**Tabla 54.** Estados del contador décadas



**Figura 124.** Estados de transición del contador de propagación MOD10

#### 5.4. Contador de propagación descendente:

Los contadores descendentes cuentan en forma inversa, por ejemplo de 1111 hasta 0000. En la Figura 125 se observa un contador descendente de 4 *bits*. Note que este contador es similar al ascendente excepto que las salidas ahora son su complemento.



**Figura 125.** Contador descendente

En la tabla 55 se muestran los estados de las salidas de los *FLIP-FLOPs*, donde se observa que después de cada pulso se decrementa la secuencia binaria representada por las salidas  $Q_3$  a  $Q_0$ .

Número de pulsos	$Q_3$	$Q_2$	$Q_1$	$Q_0$
0	1	1	1	1
1	1	1	1	0
2	1	1	0	1
...	...	...	...	...
15	0	0	0	0
16	1	1	1	1
17	1	1	1	0
...	...	...	...	...

**Tabla 55.** Estados del contador descendente

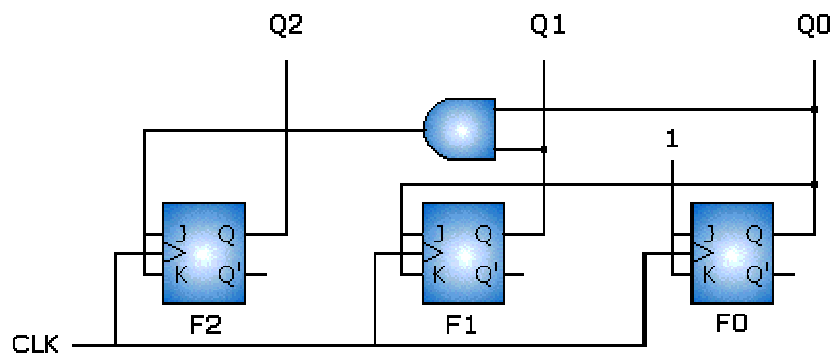
**Ejercicio:** Dibujar dos contadores binarios de 4 *bits* (ascendente y descendente) utilizando *FLIP-FLOPs* *T* que respondan al flanco negativo de la señal del reloj.

### 5.5. Contadores Sincrónicos

El inconveniente que se presenta con los contadores de propagación de la lección anterior, consiste básicamente en que no todos los *FLIP-FLOPs* cambian simultáneamente con la señal del reloj. Los contadores asincrónicos deben esperar que la señal se propague desde el primer *FLIP-FLOP* que representa el *BIT* menos significativo hasta el *FLIP-FLOP* del *BIT* más significativo.

En los contadores sincrónicos a diferencia de los contadores de propagación o asincrónicos, la señal de reloj se aplica simultáneamente a todos los *FLIP-FLOPs*. Estos contadores por lo general tienen más circuitería que los contadores de propagación y están conformados por *FLIP-FLOPs* J-K. Para entender el funcionamiento de este tipo de contadores es necesario observar con atención la secuencia para determinar los componentes que se deben agregar (generalmente *FLIP-FLOPs* y compuertas).

Analicemos el funcionamiento del contador de 3 bits que se muestra en la figura 126, y cuyos estados se resumen en la tabla 56. Asumamos que inicialmente el contador se encuentra en el estado 000. Note que el estado de la salida  $Q_0$  debe cambiar después de cada transición positiva del reloj (*CLK*), así que el *FLIP-FLOP*  $F_0$  debe tener sus entradas *J* y *K* en 1 lógico para que cumpla esta función, tal como se muestra en la figura 126.



**Figura 126.** Contador ascendente síncronico de 3 bits

Número de pulsos	$Q_2$	$Q_1$	$Q_0$
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

**Tabla 56.** Estados del contador síncronico ascendente de 3 bits

Ahora note que la salida  $Q_1$  cambia a su estado complementario cada vez que  $Q_0=1$  (ver tabla 56), así que las entradas  $J$  y  $K$  del *FLIP-FLOP*  $F1$  deben estar conectadas a la salida  $Q_0$ . De esta forma cada vez que  $Q_0=1$  y ocurra una transición positiva del reloj el *FLIP-FLOP* cambiara de estado tal como se observa en la secuencia.

Finalmente nos resta analizar el estado de la salida  $Q_2$ , para lo cual se debe observar nuevamente la tabla 56.

Note que  $Q_2$  cambia a su estado complementario cada vez que  $Q_1$  y  $Q_0$  son 1, así que la forma de implementarlo en el contador es conectado  $Q_1$  y  $Q_0$  como entradas a una compuerta *AND* y cuya salida debe ir a las entradas  $J$  y  $K$  del *FLIP-FLOP*  $F2$ .

Observe que este *FLIP-FLOP* queda en estado complemento (*toggle*), cada vez que se presente esta condición y ocurra una transición positiva en el reloj ( $CLK$ ).

Número de pulsos	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
...	...	...	...	...
15	1	1	1	1
16	0	0	0	0

**Tabla 57.** Estados del contador síncronico ascendente

Se puede hacer un análisis similar al anterior para entender el funcionamiento de este contador, sin embargo, observe que la secuencia de 3 *bits* es parte de la secuencia para 4 *bits*, así que solo basta agregar un *FLIP-FLOP*  $F3$  y una compuerta *AND* que ponga el *FLIP-FLOP*  $F3$  en modo complemento cada vez que  $Q_2$ ,  $Q_1$  y  $Q_0$  son 1, para lograr que el contador genere finalmente la secuencia de la tabla 57.

Al igual que el contador de propagación de la lección anterior, el contador síncronico se puede modificar para cambiar su número *MOD*, mediante el uso de compuertas *NAND* y las entradas asincrónicas *CLR* de los *FLIP-FLOPs*. En la Figura 127 se observa cómo se puede convertir este contador *MOD16* a *MOD10*, agregando simplemente una compuerta *NAND* de dos entradas.

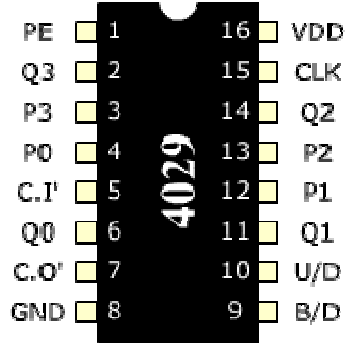
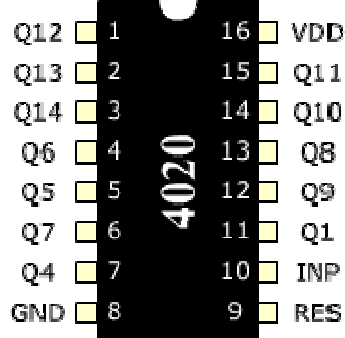
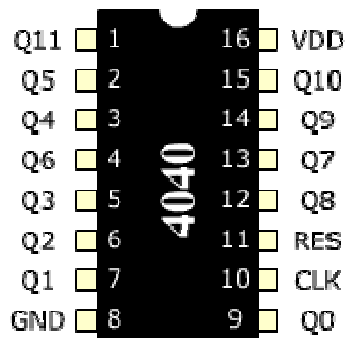
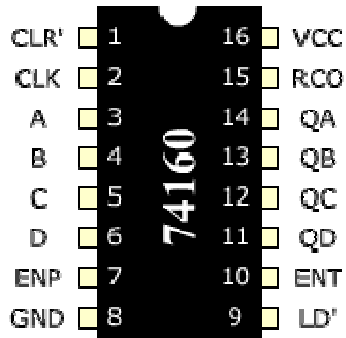


**Figura 127.** Componente adicional para convertir un contador *MOD16* a *MOD10*

### 5.6. Ejemplos de Contadores en Circuito Integrado

En el comercio existen varios contadores en circuito integrado que aparte de realizar la función de generar secuencias binarias, tiene otras funciones adicionales que generalmente tienen que ver con la configuración y modo de funcionamiento. Entre las funciones que se pueden encontrar en estos circuitos integrados se encuentran opciones de selección de secuencia ascendente o descendente, borrado así como inicialización entre otras.

A continuación se presenta una lista de algunos contadores en circuito integrado de uso difundido en Electrónica Digital, con una descripción detallada de sus pines.

Nombre	Imagen	Terminales	Descripción
Contador Binario Décadas Up/Down	 <p>Diagrama de un chip 4029, un contador binario decadal de 4 bits con entrada de reloj y salida de acarreo.</p>	PE	(Preset Enable): Esta entrada se utiliza para cargar los datos $P_i$ en las salidas $Q_i$
		$P_0...P_3$	(Presets): Entradas de fijación. Se utilizan en forma conjunta con PE
		$Q_0...Q_3$	(Quits): Salidas binarias
		C.I'	(Carry In). Entrada para detener la secuencia. En 0 Cuenta, en 1 se detiene
		C.O'	(Carry Out). Salida para indicar rebasamiento. 0 sin acarreo, 1 con acarreo
		B/D	(Bin/Dec): Selección del tipo de funcionamiento. 1 Binario, 0 decadal
		U/D	(Up/Down): Entrada de selección de secuencia. 1 Ascendente, 0 descendente
		Vdd, Gnd	Alimentación
Contador Binario (14 bits)	 <p>Diagrama de un chip 4020, un contador binario de 14 bits con entrada de reloj y salida de acarreo.</p>	$Q_1 ... Q_{13}$	(Quits): Salidas Binarias
		RES	(Reset): Entrada de Borrado
		INP	(Input): Entrada Asincrónica para incremento del contador
		Vdd, Gnd	Alimentación
Contador Binario (12 bits)	 <p>Diagrama de un chip 4040, un contador binario de 12 bits con entrada de reloj y salida de acarreo.</p>	$Q_0 ... Q_{11}$	(Quits): Salidas binarias
		CLK	(Clock): Entrada de reloj
		RES	(Reset): Entrada de Borrado
		Vdd, Gnd	(Voltaje, Ground): Alimentación
Contador Decadal (4 bits)	 <p>Diagrama de un chip 74160, un contador decadal de 4 bits con entrada de reloj y salida de acarreo.</p>	CLR'	(Clear): Entrada de borrado, en cero inicializa todas las salidas a cero
		CLK	(Clock): Entrada de Reloj
		A, B, C, D	Entradas de Datos Paralelo
		ENP, ENT	Entradas de Sostenimiento
		LD'	(Load): Entrada de habilitación de carga de datos. Se usa de forma conjunta con las entradas A, B, C, D
		QA, QB, QC, QD	(Quits): Salidas
		RCO	Salida de Rebasamiento. Cuando el contador llega al último dígito se genera un acarreo (1)



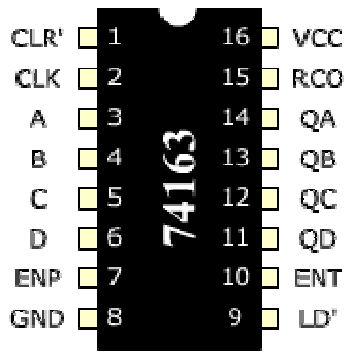
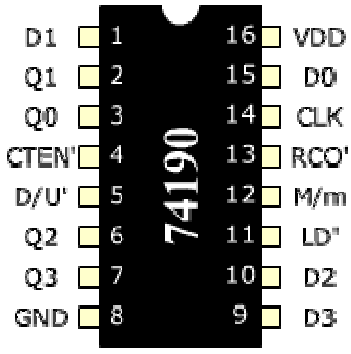
Contador Binario (4 bits)			CLR'	(Clear): Entrada de borrado, en cero inicializa todas las salidas a cero
			CLK	(Clock): Entrada de Reloj
			A, B, C, D	Entradas de Datos Paralelo
			ENP, ENT	Entradas de Sostenimiento
			LD'	(load): Entrada de habilitación de carga de datos. Se usa de forma conjunta con las entradas A, B, C, D
			QA, QB, QC, QD	(Quits): Salidas
			RCO	Salida de Rebasamiento. Cuando el contador llega al último dígito se genera un acarreo (1)
			D0, D1, D2, D3	Entradas Paralelo
Contador Decadal Up/Down (4 bits)			CTEN'	(Count Enable'): Entrada de habilitación para el contador
			D/U'	(Down/Up)': entrada de selección de la secuencia, ascendente o descendente
			Q0, Q1, Q2, Q3	(Quits): salidas binarias
			LD'	(load): entrada de carga de los datos presentes en D0...D3
			M/m	(Max/): salida de señalización de rebasamiento del contador. En 1 indica que hay carry al rebasar el número 1001 en modo ascendente o cuando alcanza el 0000 en modo descendente.
			RCO	Salida de propagación para conexión en cascada con otros contadores
			CLK	(Clock): entrada de reloj
			VDD, GND	Alimentación

Tabla 58. Contadores en Circuito Integrado

Se plantea como ejercicio, adquirir algunos de estos circuitos integrados comerciales y verificar su funcionamiento.

### 5.7. Registros de Corrimiento

En el procesamiento digital de datos se necesita con frecuencia retener los datos en ciertas ubicaciones intermedias del almacenamiento temporal, con el objeto de realizar algunas manipulaciones específicas, después de las cuales los datos modificados se pueden enviar a otra localización similar.

Los dispositivos digitales donde se tiene este almacenamiento temporal se conocen como **registros de corrimiento o registros de desplazamiento**. Dado que la memoria y el desplazamiento de información son sus características básicas, los registros son circuitos secuenciales constituidos por *FLIP-FLOPs*, donde cada uno de ellos maneja un *bit* de la palabra binaria.

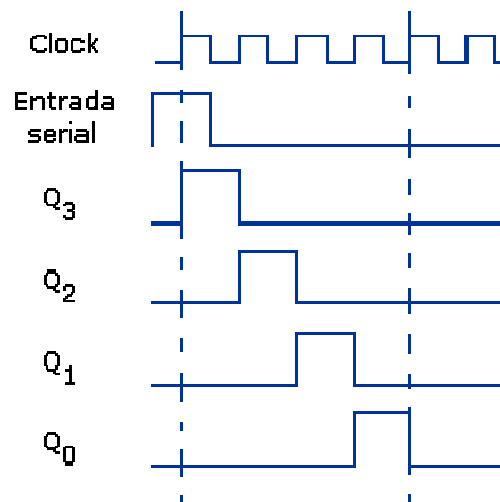
Por lo general se da el calificativo de registro a un conjunto de ocho (8) o más *FLIP-FLOPs*. Muchos registros usan *FLIP-FLOPs* tipo *D* aunque también es común el uso de *FLIP-FLOPs JK*. Ambos tipos pueden obtenerse sin dificultad como unidades

comerciales. Son muy populares los de 8 *bits*, ya que en los computadores con frecuencia manipulan *bytes* de información.

### 5.8. Registro de Corrimiento Básico

Un registro de corrimiento básico es un conjunto de *FLIP-FLOPs* conectados de tal forma que los números binarios almacenados en él son desplazados de un *FLIP-FLOP* al siguiente con cada pulso de reloj aplicado.

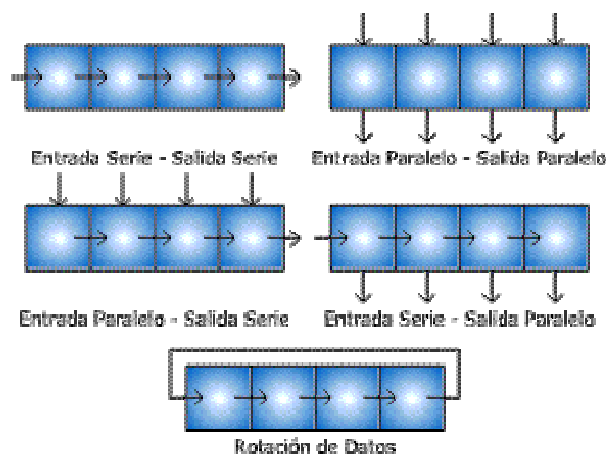
Con cada flanco ascendente del reloj la información se va desplazando hacia la derecha una posición. En la Figura 128 se observan las formas de onda de las salidas de cada *FLIP-FLOP*, donde se observa el desplazamiento de los datos de izquierda a derecha.



**Figura 128.** Formas de onda de un registro de 4 bits

### 5.9. Tipos de Entradas y Salidas en los Registros de Corrimiento

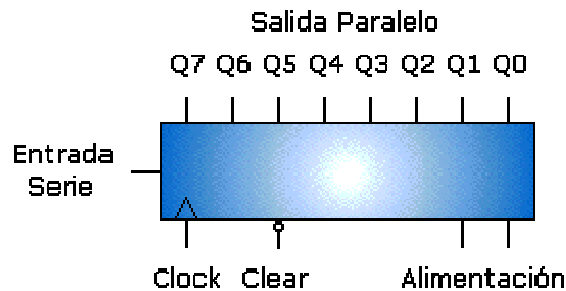
Existen diversas formas de cargar o extraer información en un registro de corrimiento. En la figura 129 se muestran las distintas formas de mover la información en un registro de corrimiento.



**Figura 129.** Tipos de Entradas y Salidas en los registros de corrimiento

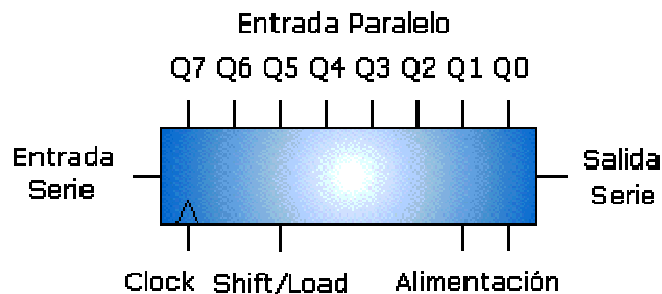
Las combinaciones de Entrada/Salida más comunes en los registros de corrimiento son: Entrada Serie/Salida Paralelo y Entrada Paralelo/Salida Serie. A continuación se dará una descripción sobre estos dos modos de funcionamiento.

**Entrada Serie - Salida Paralelo:** Es la forma más usual del tipo de entrada y salida de datos en los registros de corrimiento. En la Figura 130 se observa el esquema de un registro de esta clase. La entrada asincrónica *CLR* que se observa, es usada para poner todos los *bits* del registro en 0. Existen circuitos integrados como el *74HC164* que funcionan de esta forma.



**Figura 130.** Registro de corrimiento Entrada serie - Salida paralelo

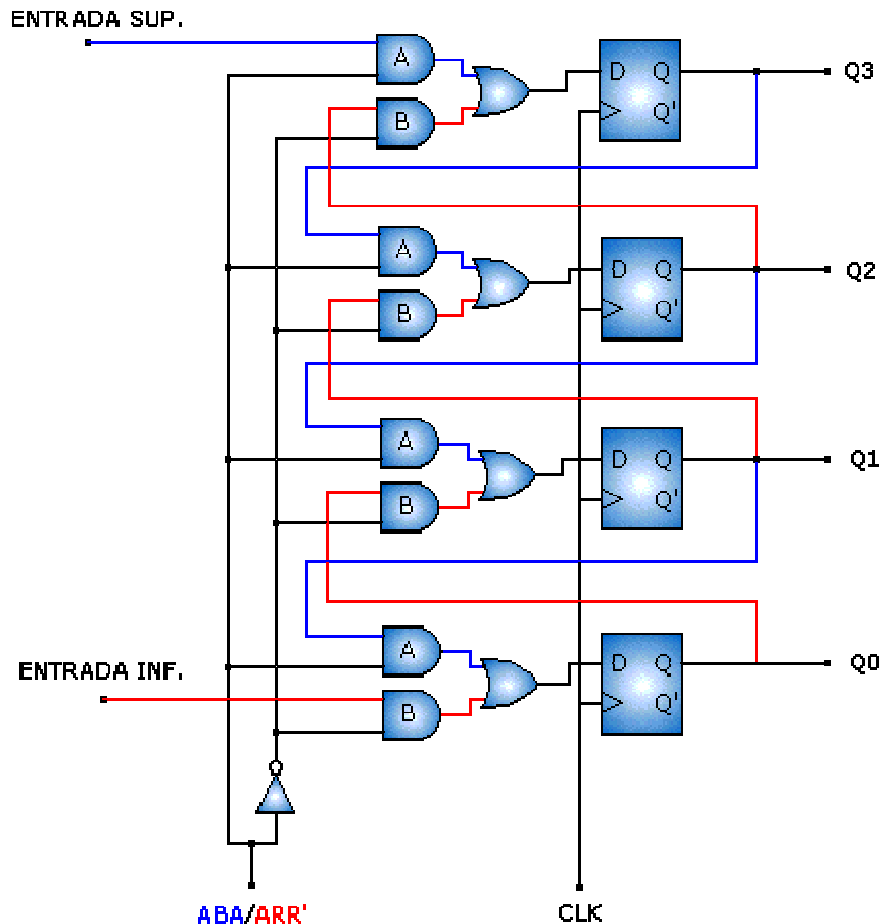
**Entrada paralelo – Salida serie:** En la Figura 131 se observa el esquema de un registro de este tipo. *LOAD*: Las entradas en paralelo se almacenan en los *FLIP-FLOPs* internos (entrada asincrónica), *SHIFT*: Corrimiento del puerto hacia la derecha (entrada sincrónica), entrada serie por el primer *FLIP-FLOP* y salida serial por el último. Existen circuitos integrados como el *74HC165* que funcionan con base en este esquema.



**Figura 131.** Registro de corrimiento Entrada paralelo - Salida serie

### 5.10. Registros de corrimiento bidireccionales

Este tipo de registro tiene la opción de elegir la dirección en que se transmiten los datos. Estos registros tienen una señal de control que permite seleccionar el sentido de desplazamiento de los datos. En la Figura 132 se observa el circuito lógico de un registro bidireccional de 4 *bits*.



**Figura 132.** Registro de corrimiento bidireccional de 4 bits

Para propósitos de entender el funcionamiento de este registro se ha dispuesto de forma vertical, para mostrar cómo se desplazan los datos. Cuando la entrada *ABA/ARR'* se encuentra en 1 lógico, los datos se desplazan hacia abajo y cuando esta es 0 lógico los datos se desplazan hacia arriba.

Cuando la señal de control *ABA/ARR'* es 1, las compuertas marcadas con *A* se activan, permitiendo que el dato de cualquier *FLIP-FLOP* pase al *FLIP-FLOP* inmediatamente inferior después de que ocurra una transición positiva en la señal del reloj, de esta forma la información se desplaza por las líneas marcadas en azul que se observan en la figura 132.

Cuando la señal de control *ABA/ARR'* es 0, las compuertas marcadas con *B* se activan y el dato de cualquier *FLIP-FLOP* se pasa al *FLIP-FLOP* inmediatamente superior. Las líneas marcadas en rojo en la figura 132 indican el canal de transmisión de los datos de un *FLIP-FLOP* a otro para esta condición.

Note que las compuertas marcadas como *A* y *B* se activan de forma complementaria, es decir, mientras se activan aquellas marcadas como *A* las marcadas como *B* se encuentran inactivas y viceversa.

### 5.11. Registros en Circuito Integrado

En el mercado existen actualmente varios circuitos integrados que desempeñan su función como registros, en esta sección mencionaremos algunos de estos registros disponibles en lógica *TTL* y *CMOS*.

### Circuito Integrado 74HC373

Este integrado contiene 8 *Cerrojos* tipo *D* con salidas triestado. En la figura 133 se observa el esquema de conexiones interno y la descripción de sus entradas y salidas es la siguiente:

- $D0...D7$ : Entrada paralelo
- $Q0...Q7$ : Salida paralelo
- $LE$ : Latch Enable
- $OE$ : Output Enable

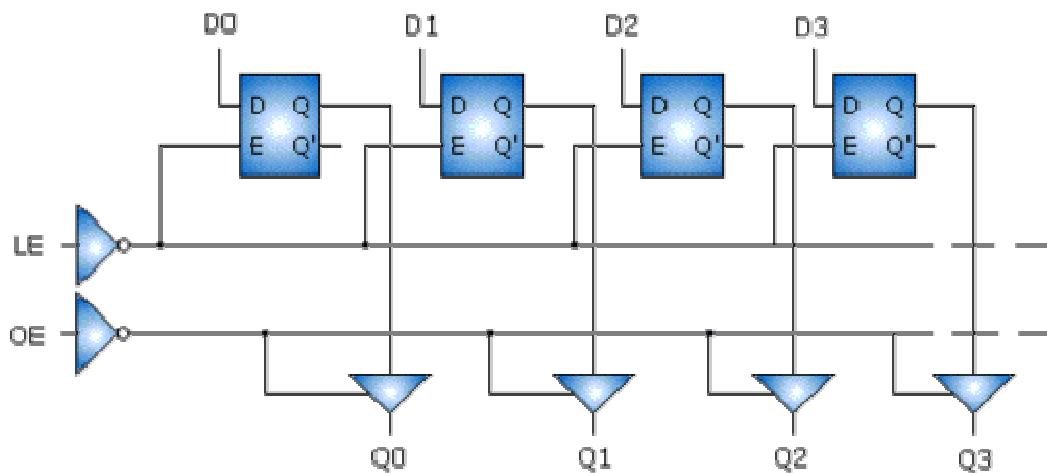
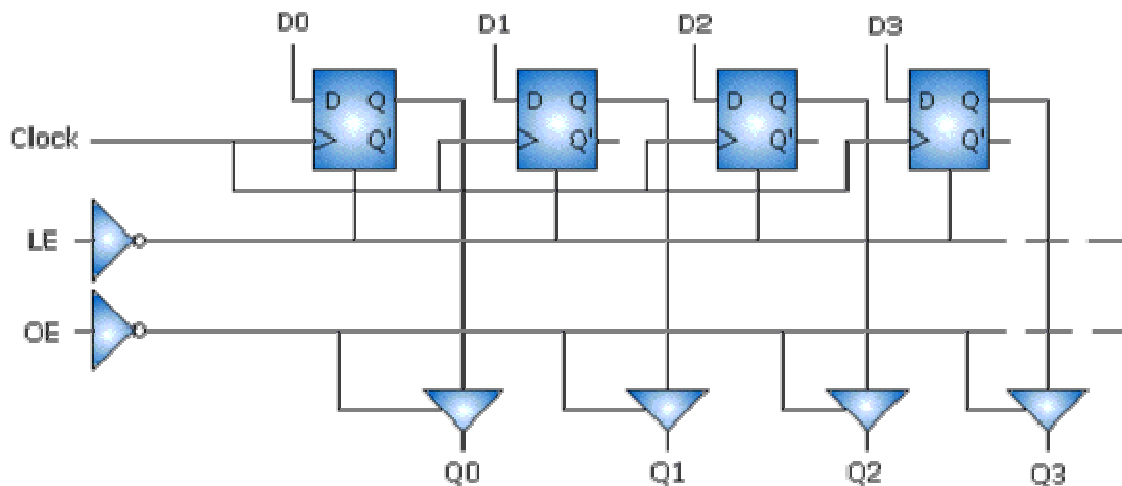


Figura 133. Diagrama Lógico del CI 74HC373

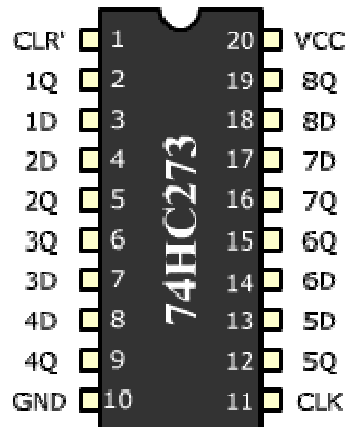
### Circuito Integrado 74HC374

Este circuito integrado contiene 8 *FLIP-FLOPs* tipo *D* con salidas triestado sensibles al flanco de subida de la señal del Reloj. En la figura 134 se muestra la estructura interna de este registro y su diferencia con el anterior Circuito Integrado es que este contiene *FLIP-FLOPs*.



**Figura 134.** Diagrama lógico del CI74HC374**Circuito Integrado 74HC273**

Este integrado contiene 8 *FLIP-FLOPs* tipo *D* con salidas triestado sensibles al flanco de subida de la señal del reloj, adicionalmente tiene una entrada para borrar activa en bajo (*CLR'*). En la figura 135 se observa el diagrama de pines de este integrado y el tabla 59 los estados lógicos.

**Figura 135.** Esquema del CI-74HC273

CLEAR	CLK	D	Q
0	X	X	0
1	↑	1	1
1	↑	0	0
1	0	X	Q0

**Tabla 59.** Descripción de las entradas del CI-74HC273**5.12. Aplicaciones de los Registros de Corrimiento**

Los registros de corrimiento tienen varias aplicaciones en la Electrónica Digital, entre las cuales se pueden mencionar las siguientes:

- Transmisión de datos.
- Conversión de protocolo serie en paralelo y viceversa.
- Puertos de salida de los microcomputadores.
- Secuenciadores (luces y anuncios publicitarios).
- Multiplicaciones y divisiones por 2, 4, 8, 16 *bits*.
- Operaciones que se hacen en forma secuencial.

**5.13. Contador en Anillo**

El contador en anillo es un registro de corrimiento básico en el que los datos no se pierden al desplazarse, en lugar de ello, la información rota debido a que los *FLIP-FLOPs* de los extremos se encuentran interconectados, de tal forma que los datos se desplazan en forma de "anillo".

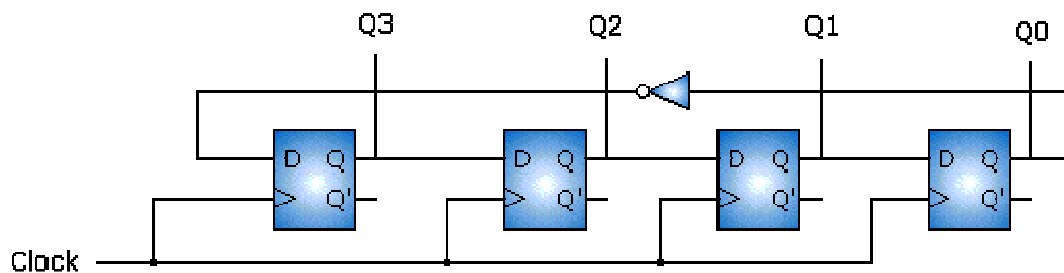
Asumiendo que el estado inicial del contador en anillo es 1000 ( $Q_3=1$ ,  $Q_2=0$ ,  $Q_1=0$ ,  $Q_0=0$ ), los estados que se presentarían en este contador serían los mostrados en la tabla 60. Después del cuarto pulso en la señal del reloj el estado inicial se repite.

Pulso del Reloj	Q3	Q2	Q1	Q0
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

**Tabla 60.** Estados del Contador en Anillo

En el mercado existen contadores de este tipo en circuito integrado, sin embargo su construcción es muy fácil a partir de un registro de corrimiento convencional.

Existe otro contador en anillo llamado contador Johnson, el cual tiene un funcionamiento similar al contador en anillo, excepto que el estado del último *FLIP-FLOP* se realimenta al primero a través de un inversor. En la figura 136 se observa el diagrama lógico de este contador.



**Figura 136.** Contador Johnson de 4 bits

Tomando como estado inicial del contador Johnson 0000 ( $Q_3=0$ ,  $Q_2=0$ ,  $Q_1=0$ ,  $Q_0=0$ ), los estados presentes en este contador serían los mostrados en la tabla 61. Note que durante el octavo pulso en la señal del reloj el estado inicial se repite.

Pulso del Reloj	Q3	Q2	Q1	Q0
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

**Tabla 61.** Estados del Contador Johnson

## 6. Análisis y Diseño de Circuitos Secuenciales

El Análisis y Diseño de Circuitos Secuenciales se encuentra estrechamente relacionado con el control secuencial, denominado también control lógico o control binario. En los sistemas de control secuencial las entradas y las salidas son de tipo binario y determinan una serie de pasos para la operación de un proceso.

Las entradas por lo general son: pulsadores, interruptores, microinterruptores, fines de carrera o detectores de proximidad. Las salidas pueden ser: Válvulas solenoides, cilindros neumáticos, contactores para arranque y parada de motores, pilotos de señalización, alarmas, entre otros.

Cuando el sistema de control secuencial es pequeño se realiza con circuitos digitales combinatorios y secuenciales. Cuando es grande se realiza con *PLC's* (Controladores Lógicos Programables), microcomputadores, microprocesadores especiales para control secuencial y por software en *PC*.

En este capítulo se mostrarán las metodologías básicas para el *Diseño de Circuitos Secuenciales* y su aplicabilidad en dispositivos secuenciales para funciones específicas.

### 6.1. Teoría de Máquinas de Estado (FSM)

La teoría de máquinas de estado es el nombre con el que se conocen los métodos de Análisis y Diseño de Circuitos Secuenciales Sincrónicos. Esta lección constituye una introducción al tema del capítulo, donde se definirá lo que son las máquinas de estado y los conceptos básicos para entender la metodología de Análisis y Diseño de Circuitos Secuenciales.

Las máquinas de estado son circuitos secuenciales que se encuentran constituidos por una etapa combinatorial y una etapa de memoria, relacionadas de tal forma que conforman un sistema secuencial para algún propósito especial. Los registros y contadores con entradas asincrónicas son ejemplos de este tipo de sistemas secuenciales.

### 6.2. Máquinas de Estado de Mealy y Moore

Los circuitos secuenciales se clasifican dentro de una categoría conocida como máquinas de estado, de la cual se distinguen comúnmente dos tipos:

- **Máquina de Mealy:** En esta máquina de estados las salidas se encuentran determinadas por el estado interno del sistema y por las entradas no sincronizadas con el circuito. El diagrama de bloques representativo de esta máquina se muestra en la figura 137, donde se observa que las salidas del sistema son tanto sincrónicas como asincrónicas.



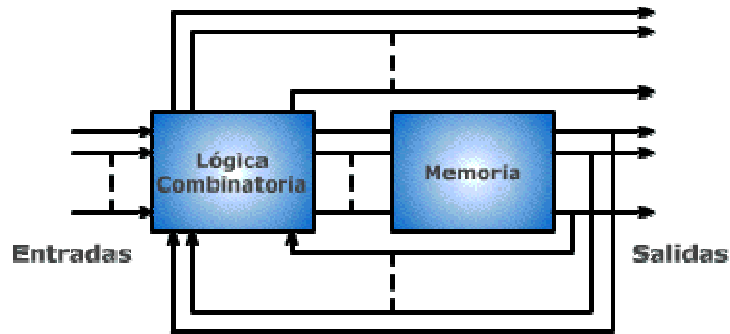


Figura 137. Máquina de estados de Mealy

- **Máquina de Moore:** Las salidas solo dependen del estado interno y de cualquier entrada sincronizada con el circuito, como se observa en la figura 138, donde las salidas del sistema son únicamente sincrónicas. Un ejemplo de este tipo de máquinas de estado son los contadores (ver capítulo 5).

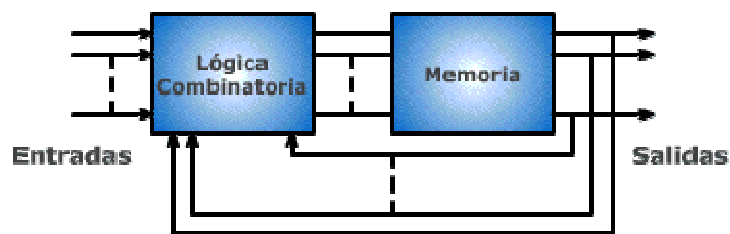


Figura 138. Máquina de estados de Moore

Los circuitos secuenciales se caracterizan por tener una etapa combinacional y otra de memoria conformada por *FLIP-FLOPs*. En la figura 140, se puede observar un ejemplo particular de este tipo de circuitos, el cual corresponde a una Máquina de estado de **Mealy**. Observe que hay salidas que dependen de la etapa de memoria y hay una salida que depende directamente de la etapa combinacional.

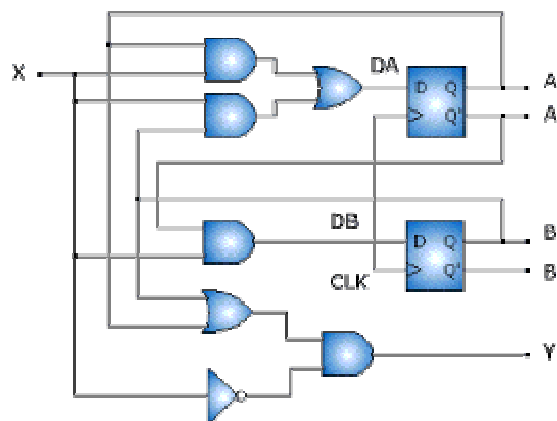


Figura 140. Circuito Secuencial de ejemplo

Con base en el circuito de la figura 140, se dará una descripción de las herramientas básicas que son empleadas para el Análisis y Diseño de Circuitos Secuenciales. Entre estas herramientas se encuentran las ecuaciones lógicas, los diagramas de estado, las tablas de estado, las tablas de transición y los mapas de *Karnaugh*.

### 6.3. Ecuaciones Lógicas

Las ecuaciones lógicas son funciones que definen la relación existente entre los estados de entrada y los estados de salida del sistema. Para determinar las ecuaciones lógicas de la máquina de estados de la figura 140, inicialmente se deben identificar los estados siguientes. Estos estados corresponden a aquellos que ocurren después de una transición en la señal de reloj de los *FLIP-FLOPs*. Recuerde que para los *FLIP-FLOPs* tipo *D* el estado siguiente ( $Q_{i+1}$ ) es igual al estado de la entrada *D*. Teniendo en cuenta lo anterior las ecuaciones lógicas para los *FLIP-FLOPs* *A* y *B* del circuito de la figura 140 serían las siguientes:

$$A = D_A = A \cdot X + B \cdot X$$

$$B = D_B = A' \cdot X$$

La salida *Y* esta dada por:

$$Y = (A + B) \cdot X'$$

Observando esta última ecuación se concluye que la salida (*Y*) es función del estado presente del sistema (*A* y *B*) y de la entrada asincrónica (*X*).

Las ecuaciones lógicas en los circuitos secuenciales tienen una estructura formada por dos clases de estados:

- Los estados siguientes, los cuales se agrupan al lado izquierdo de la expresión y representan las variables dependientes del sistema. El estado de estas variables cambia en el momento que ocurra una transición en la señal de reloj.
- Los estados actuales y entradas del sistema. Agrupados al lado derecho de la expresión, constituyen las variables independientes, las cuales pueden o no cambiar en sincronía con el sistema.

Cuando las ecuaciones de estado contienen varios términos, se pueden simplificar empleando metodologías de reducción de términos como Álgebra de Boole, Mapas de Karnaugh, o mediante el Algoritmo de Quine-McCluskey, las cuales fueron presentadas en el Capítulo 2.

### 6.4. Tablas de Estado

Una tabla de estado es un listado que contiene la secuencia de los estados de entradas, estados internos y salidas del sistema, considerando todas las posibles combinaciones de estados actuales y entradas. Las tablas de estado por lo general se dividen en tres partes: estados actuales, estados siguientes y salidas, tal como se muestra en la tabla 62.

Estados actuales		Entrada	Estados siguientes		Salida
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	0	1

**Tabla 62.** Tabla de estado (circuito Figura 140)

La tabla de estado para un circuito secuencial con  $m$  *FLIP-FLOPs* y  $n$  entradas tiene  $2^{m+n}$  filas. El estado siguiente tiene  $m$  columnas, y el número de columnas depende del número de salidas.

Existe una forma más conveniente de organizar la información en la tabla de estado, la cual se muestra en la Tabla 63, donde los estados se agrupan de tal modo que la tabla se puede traducir a un diagrama de estados. Al igual que la tabla anterior esta tiene tres secciones: estados actuales, estados siguientes y salidas, sin embargo los estados se agrupan dependiendo del valor de las entradas. La sección de estados actuales agrupa los estados que ocurren antes de una transición en la señal de reloj, la sección de estados siguientes lista aquellos que ocurren después de la transición del reloj y la sección de salidas reúne los estados que se dan en el mismo instante de los estados actuales.

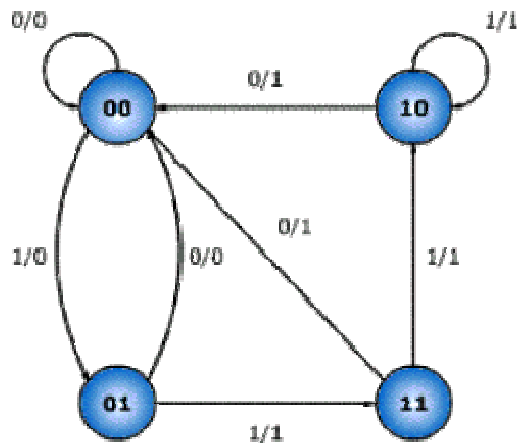
Estado Actual	Estado Siguiente		Salida	
	X=0	X=1	X=0	X=1
AB	AB	AB	Y	Y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

**Tabla 63.** Tabla de estado (forma simplificada)

Haciendo un análisis de la operación del circuito de la figura 140, se puede observar lo siguiente: Cuando la variable  $X=0$  los estados actuales  $A$  y  $B$  cambian a 0 después de la transición de reloj, y cuando  $X=1$ , los estados de las salidas se comportan tal como se resume en la tabla 63. Se plantea como ejercicio verificar la información de la tabla.

### 6.5. Diagramas de Estado

Un diagrama de estados es una representación gráfica que indica la secuencia de los estados que se presentan en un circuito secuencial, teniendo en cuenta las entradas y salidas. El diagrama se forma con círculos y líneas. Los círculos representan los estados del circuito secuencial y cada uno de ellos contiene un número que identifica su estado. Las líneas indican las transiciones entre estados y se marcan con dos números separados por un (/), estos dos números corresponden a la entrada y salida presentes antes de la transición. A manera de ejemplo observe la línea que une los estados 00 y 01 en el diagrama de estado de la figura 141. Esta línea marcada como 1/0 indica que el circuito secuencial se encuentra en el estado 00 mientras la entrada  $X=0$  y la salida  $Y=0$ , y que después de que ocurra una transición en la señal de reloj el estado cambia a 01.



**Figura 141.** Diagrama de estados correspondiente a la Tabla 63

Las líneas que salen y regresan al mismo círculo indican que no hay cambio en el estado, cuando se presentan la entrada y salida indicados.

### 6.6. Tablas de Transición de FLIP-FLOPs

Las tablas de transición se usan en conjunto con las de estado y representan la tabla de verdad de los *FLIP-FLOPs* con los cuales se desea implementar el circuito secuencial. La tabla contiene los estados actuales y siguientes según el estado de las entradas de los *FLIP-FLOPs*. La tabla 64, corresponde a la tabla de transición del *FLIP-FLOP JK*.

Transiciones de Salida		Entradas al FLIP-FLOP	
$Q_i$	$Q_{i+1}$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

**Tabla 64.** Tabla de transición del FLIP-FLOP JK

En la tabla,  $Q_i$  corresponde al estado actual y  $Q_{i+1}$  al estado siguiente,  $J$  y  $K$  son las entradas de los *FLIP-FLOPs*. La información sombreada en la tabla se interpreta de la siguiente forma: cuando el estado presente de la salida  $Q=0$  y las entradas  $J=1$  y  $K=X$  ( $X$  indica una condición de no importa, 1 o 0), después de un pulso de reloj en el *FLIP-FLOP* la salida cambia al estado siguiente  $Q=1$ .

### 6.7. Mapas de Karnaugh

Generalmente las tablas de estado y de transición de los *FLIP-FLOPs* se fusionan en una sola para agrupar la información de tal forma que permitan construir los *Mapas de Karnaugh* para simplificar las funciones lógicas. La tabla 65 corresponde a una tabla de estado de un contador de tres bits con *FLIP-FLOPs JK*. Observe que esta tabla incluye las entradas  $J$  y  $K$  para cada una de la transiciones (estado actual a estado siguiente). Las regiones sombreadas en la tabla indican que el estado  $Q_i$  cambia estando presentes las entradas  $J_i$  y  $K_i$  correspondientes después de una transición del reloj.

Estado Actual			Estado Siguiente			Entradas de los FLIP-FLOP					
Q2	Q1	Q0	Q2	Q1	Q0	J2	K2	J1	K1	J0	K0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

**Tabla 65.** Tabla de estado y transición de un contador de 3 bits

Los *Mapas de Karnaugh* se emplean para definir la lógica de las entradas de los *FLIP-FLOPs* y se debe hacer uno para cada una de las entradas. La figura 142 corresponde al *Mapa de karnaugh* de la entrada  $J_1$ , de la tabla de estado 65.

$Q_1 Q_0$		00	01	11	10
$Q_2$	0	0	1	X	X
	1	0	1	X	X

**Figura 142.** Mapa de Karnaugh para el estado  $J_1$

Observe que cada celda en el mapa representa uno de los estados actuales de la secuencia en la tabla de estado. Una vez asignados todos los estados posibles a cada celda en el *Mapa de Karnaugh* se procede a simplificar y deducir las expresiones lógicas. En la figura 142 se observa que la expresión correspondiente a la entrada  $J_1$  es:

$$J_1 = Q_0$$

Esta expresión indica que en el circuito lógico la salida  $Q_0$  debe ir conectada a la entrada  $J_1$ . En la siguiente lección se explicará de una forma detallada el procedimiento para el Diseño de Circuitos Secuenciales.

### 6.8. Análisis y Diseño de Circuitos Secuenciales Sincrónicos

La gran mayoría de los circuitos digitales contienen *FLIP-FLOPs* y compuertas para realizar funciones específicas. El diseño de estos circuitos inicia a partir de las especificaciones y finaliza con las funciones lógicas, de las cuales se obtiene el circuito lógico.

Inicialmente se debe crear una tabla de estado o representación equivalente, para identificar la secuencia de estados que deseada. Luego de seleccionar el número y tipo de *FLIP-FLOPs* con los cuales se desea hacer el diseño, se deduce la lógica combinatoria necesaria para generar la secuencia de estados.

Los circuitos secuenciales se pueden analizar y diseñar siguiendo un procedimiento claramente definido que consiste en los siguientes pasos:

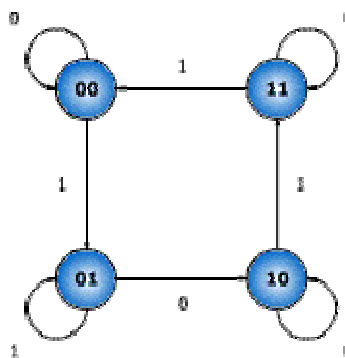
1. Asignación de estados
2. Construcción del diagrama de transición
3. Elaboración de la tabla de estados
4. Obtención de ecuaciones o funciones lógicas
5. Realización de circuitos lógicos

Para explicar este método se desarrollará un ejemplo aplicado a un diseño particular.

**Ejemplo 1:** Diseñar el circuito secuencial del proceso que se cumple de acuerdo al diagrama de estados de la figura 143.

#### Paso 1. Asignación de estados

Este proceso tiene cuatro estados, una entrada y no tiene salidas (se pueden considerar como salidas las de los *FLIP-FLOPs*). Para representar los cuatro estados se usarán dos *FLIP-FLOPs* identificados como  $A$  y  $B$  de tipo  $JK$  y la entrada será identificada como  $X$ .



**Figura 143.** Diagrama de estados

Paso 2. Construcción del diagrama de la transición o de estado

La figura 143 corresponde al diagrama de transición. Analizando este diagrama se observa que el estado 10 se mantiene mientras  $X=0$  y en el momento que  $X=1$  pasa al estado 11, después al estado 00 y finalmente al estado 01, hasta el momento que nuevamente  $X=0$ , volviendo de esta forma al estado  $AB=10$ . Adicionalmente observe que los estados 00 10 y 11, se mantienen cuando  $X=0$  y el estado 01 se mantiene cuando  $X=1$ .

Paso 3. Elaboración de la tabla de estados

A partir del diagrama de estados y de la tabla de transición del *FLIP-FLOP JK* se puede construir la tabla de estados (ver tabla 66).

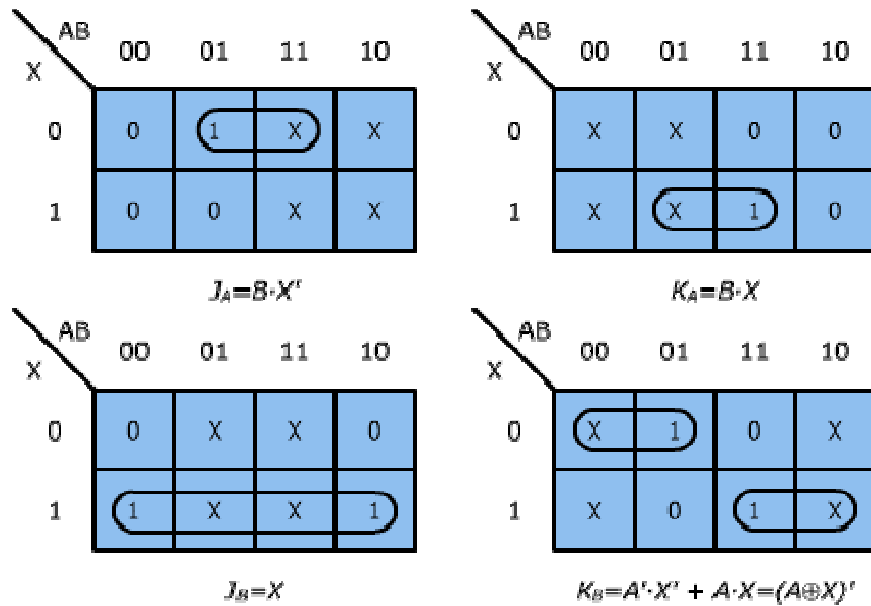
Entrada	Estado Actual		Estado Siguiente		Excitaciones			
X	A	B	A	B	$J_A$	$K_A$	$J_B$	$K_B$
0	0	0	0	0	0	X	0	X
1	0	0	0	1	0	X	1	X
0	0	1	1	0	1	X	X	1
1	0	1	0	1	0	X	X	0
0	1	0	1	0	X	0	0	X
1	1	0	1	1	X	0	1	X
0	1	1	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

**Tabla 66.** Tabla de estado

Para la simplificación de los circuitos combinatorios es conveniente que se presenten condiciones de "no importa", ya que estas, permiten simplificar las funciones lógicas y por tanto el tamaño del circuito lógico.

Paso 4. Obtención de ecuaciones o funciones lógicas.

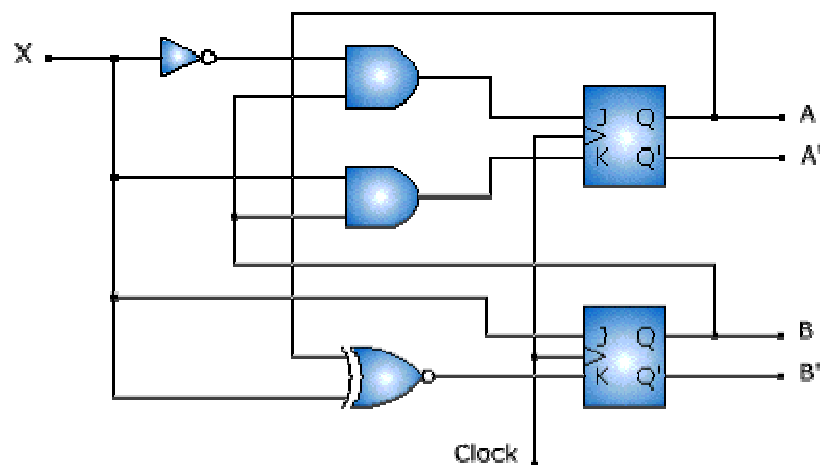
En este paso se obtienen las funciones lógicas para las entradas de los *FLIP-FLOPs* ( $J_A$ ,  $K_A$ ,  $J_B$  y  $K_B$ ) y el objetivo es deducir la lógica combinatoria de estado siguiente, mediante el uso de *Mapas de Karnaugh*. A continuación en la figura 144 se muestran los *Mapas de Karnaugh* y las funciones lógicas correspondientes.



**Figura 144.** Mapas de Karnaugh para las entradas  $J_A$ ,  $J_B$ ,  $K_A$  y  $K_B$

### Paso 5. Realización de circuitos lógicos

Este es el ultimo paso del diseño, y consiste en implementar la lógica combinacional a partir de las ecuaciones lógicas obtenidas en el paso anterior para las entradas  $J$  y  $K$  de los *FLIP-FLOPs*. Las conexiones correspondientes, se efectúan mediante el uso de compuertas e inversores y en la figura 145 se muestra el diseño final del circuito lógico.



**Figura 145. Circuito Lógico del Diseño**



### 6.9. Diseño de Circuitos Secuenciales con *FLIP-FLOPs D*

El diseño del circuito de la figura 145 se hizo con *FLIP-FLOPs JK*. En esta sección veremos como se realiza el diseño de circuitos secuenciales mediante el uso de *FLIP-FLOPs* tipo *D*.

A diferencia de las entradas de los *FLIP-FLOPs JK*, las entradas en los *FLIP-FLOPs D* corresponden exactamente a los estados siguientes. Por esta razón en la tabla de estado no se requiere una columna independiente para las excitaciones. En el siguiente ejemplo se verá como realizar el diseño de circuitos secuenciales con *FLIP-FLOPs D*.

**Ejemplo 2:** Realizar el diseño del circuito lógico correspondiente a la tabla de estado 67. Observe que esta tabla es la misma del ejemplo anterior, pero adicionalmente se agregó una salida (*Y*).

Entrada	Estado actual		Estado siguiente		Salida
X	A	B	A(D <sub>A</sub> )	B(D <sub>B</sub> )	Y
0	0	0	0	0	0
1	0	0	0	1	1
0	0	1	1	0	0
1	0	1	0	1	0
0	1	0	1	0	0
1	1	0	1	1	1
0	1	1	1	1	0
1	1	1	0	0	0

**Tabla 67.** Tabla de estado

#### Paso 1. Asignación de estados

Este proceso al igual que el ejemplo anterior tiene cuatro estados de dos *bits* (*AB*), una entrada (*X*) y una salida (*Y*). Para representar los cuatro estados se usarán dos *FLIP-FLOPs D* identificados como *A* y *B*.

#### Paso 2. Construcción del diagrama de la transición o de estado

El diagrama de transición es el mismo del ejemplo anterior, excepto que ahora se tiene en cuenta la salida (*Y*). En la figura 146 se observa el diagrama de estado.

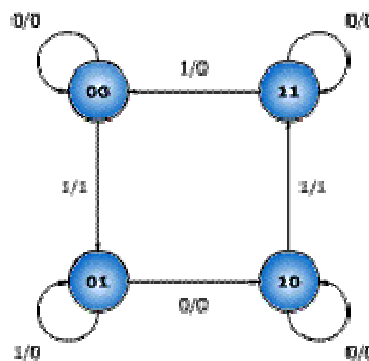


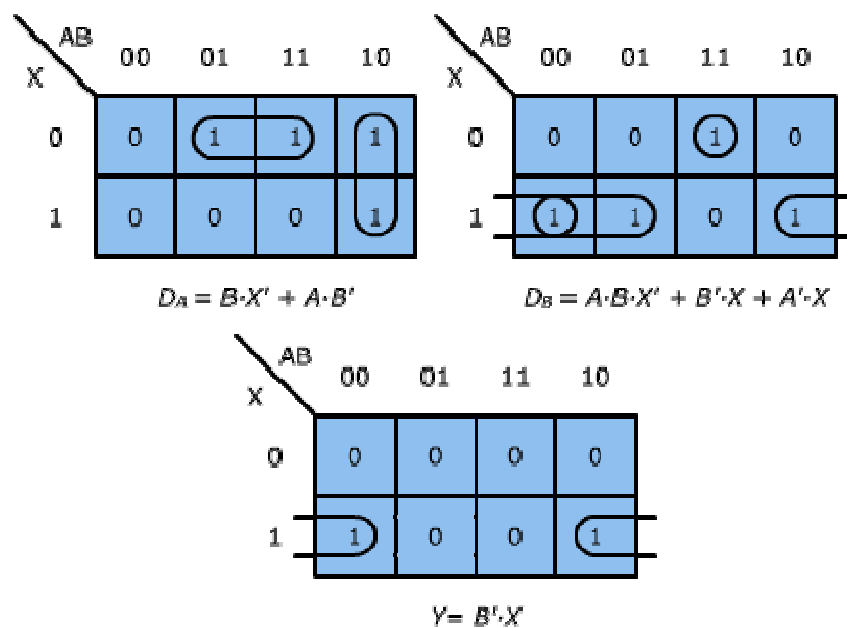
Figura 146. Diagrama de estados - Ejemplo 2

Paso 3. Elaboración de la tabla de estado.

Para este ejemplo inicialmente se dió la tabla de estados, la cual se observa en la tabla 67.

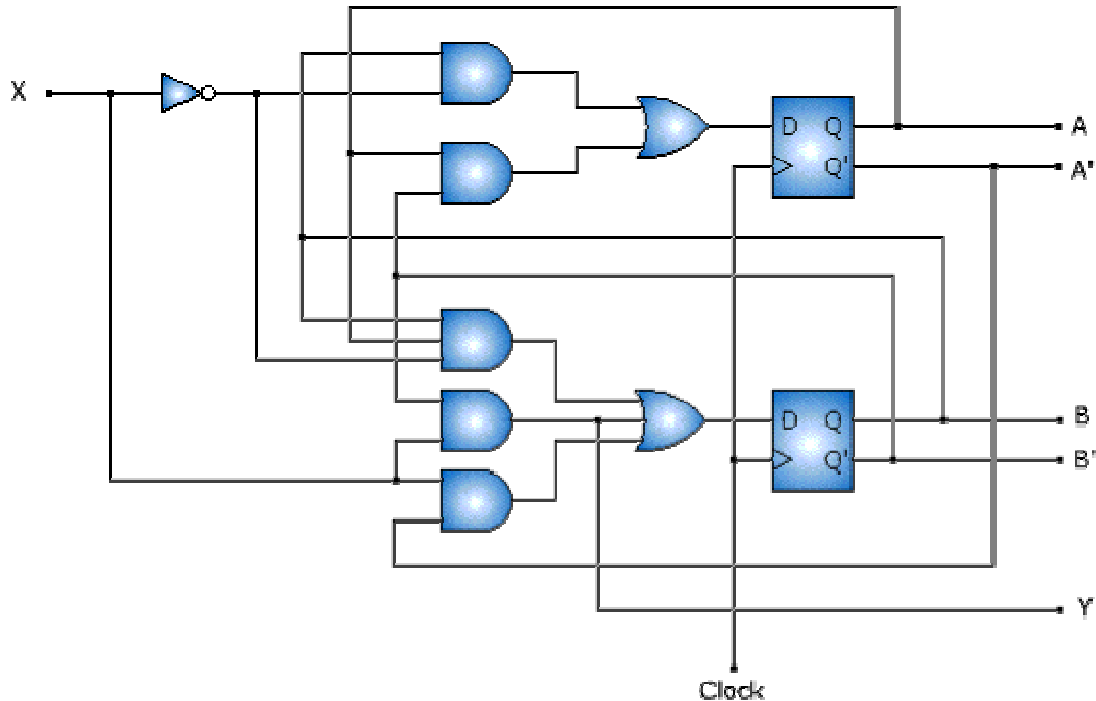
Paso 4. Obtención de ecuaciones o funciones lógicas.

En este paso se deben obtener las funciones lógicas para las entradas de los *FLIP-FLOPs* ( $D_A$ ,  $D_B$ ) y la salida ( $Y$ ). En la figura 147 se muestran los *Mapas de Karnaugh* y las funciones lógicas correspondientes.

**Figura 147.** Mapas de Karnaugh para las entradas  $D_A$ ,  $D_B$ , y  $Y$

Paso 5. Realización de circuitos lógicos

Con las ecuaciones lógicas obtenidas en el paso anterior se puede implementar el circuito lógico. Las conexiones correspondientes, se efectúan mediante el uso de compuertas e inversores y en la figura 148 se muestra el diseño del circuito.



**Figura 148.** Circuito Lógico

### 6.10. Estados no usados

Durante el diseño de los circuitos secuenciales para simplificar las representaciones lógicas, es conveniente emplear los estados no usados como condiciones que no importa. Estos estados se identifican con una (X) en los *Mapas de Karnaugh*.

Para ilustrar como emplear estos estados, observe la tabla 68.

Teniendo en cuenta todas las posibles combinaciones de las variables  $A$ ,  $B$ ,  $C$  y  $X$ , Note que en esta tabla hay seis estados que no están presentes (0000, 0001, 1100, 1101, 1110 y 1111).

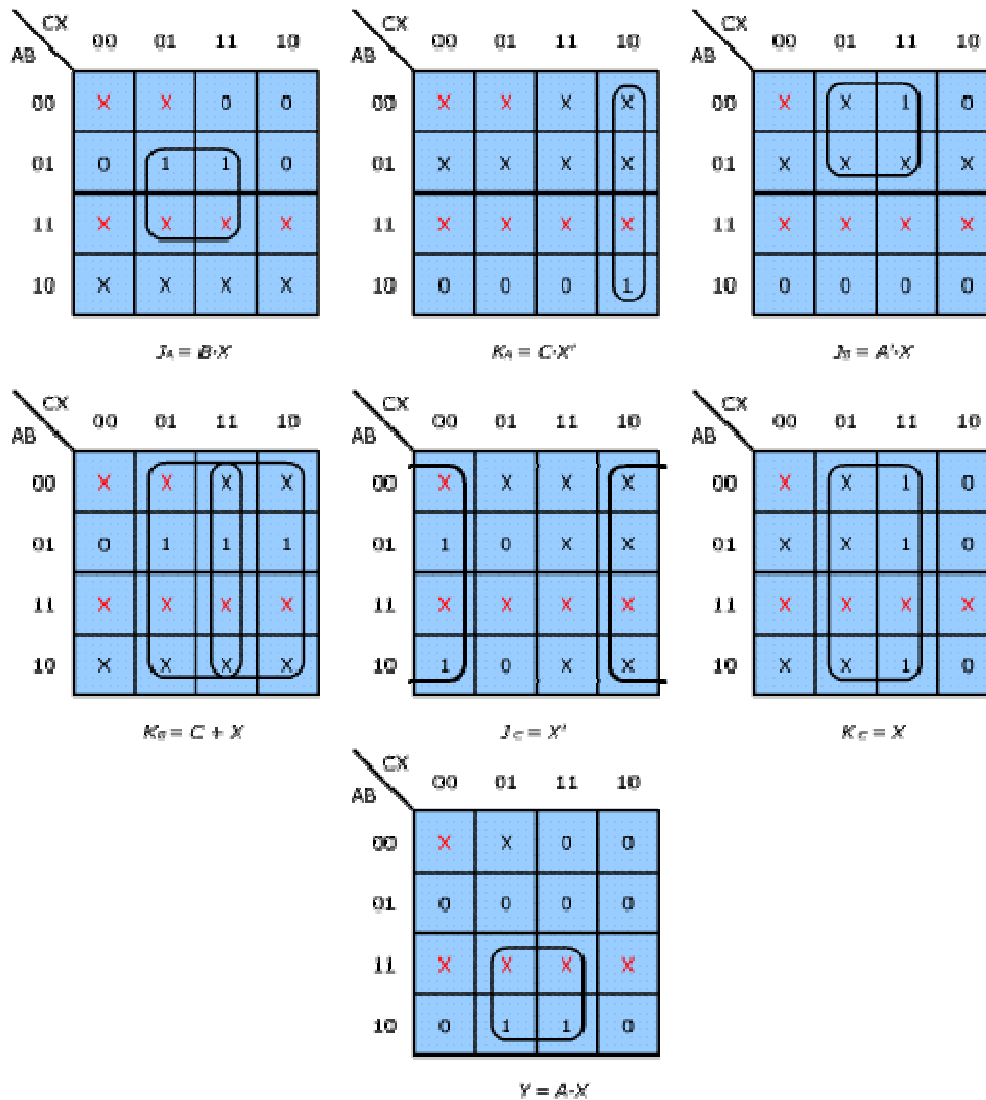
Las seis filas de la tabla correspondientes a estos estados se identifican como estados  $X$  (1 o 0) o condiciones de "No importa", al momento de elaborar los *Mapas de Karnaugh*.

Estado actual			Entrada	Estado siguiente			Excitaciones						Salida
A	B	C	X	A	B	C	J <sub>A</sub>	K <sub>A</sub>	J <sub>B</sub>	K <sub>B</sub>	J <sub>C</sub>	K <sub>C</sub>	Y
0	0	1	0	0	0	1	0	X	0	X	X	0	0
0	0	1	1	0	1	0	0	X	1	X	X	1	0
0	1	0	0	0	1	1	0	X	X	0	1	X	0
0	1	0	1	1	0	0	1	X	X	1	0	X	0
0	1	1	0	0	0	1	0	X	X	1	X	0	0
0	1	1	1	1	0	0	1	X	X	1	X	1	0
1	0	0	0	1	0	1	X	0	0	X	1	X	0
1	0	0	1	1	0	0	X	0	0	X	0	X	1
1	0	1	0	0	0	1	X	1	0	X	X	0	0
1	0	1	1	1	0	0	X	0	0	X	X	1	1

**Tabla 68.** Tabla de estado - Condiciones de "no importa"

Los *mapas de karnaugh* correspondientes a las entradas de cada *FLIP-FLOP* ( $J_A$ ,  $K_A$ ,  $J_B$ ,  $K_B$ ,  $J_C$  y  $K_C$ ) y la salida ( $Y$ ), se muestran en la figura 149.

Observe que en cada mapa los estados resaltados en rojo corresponden a los estados no usados, los cuales se han incluido como condiciones "no importa" para simplificar la mayor cantidad de variables en las expresiones.



**Figura 149.** Mapas de Karnaugh para las entradas a los FLIP-FLOPs

Como conclusión sobre esta sección, podemos decir que es recomendable incluir los estados no usados en el diseño de los circuitos secuenciales. Esto implica una reducción en las expresiones lógicas y por consiguiente en el tamaño del circuito, que en otros términos representará obviamente un menor tiempo de desarrollo y costo de implementación.

Se plantea como ejercicio hacer el diagrama lógico correspondiente a las ecuaciones halladas a partir de los mapas de *Karnaugh* de la figura 149 y hacer el diseño del circuito secuencial sin tener en cuenta los estados no usados para comparar los dos casos y notar las diferencias.

Para más información sobre simplificación de funciones lógicas ver El Capítulo 2, numeral 2.6. Métodos para Sintetizar Circuitos Lógicos, literal c. Mapas de Karnaugh.

## 6.11. Análisis de Circuitos Secuenciales Asincrónicos

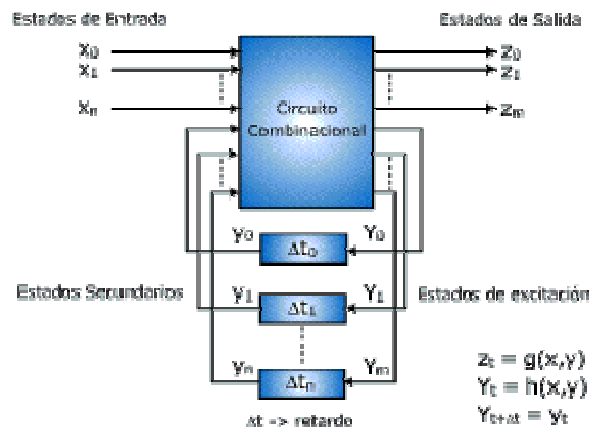
El análisis de Circuitos Asincrónicos es similar al análisis de los circuitos sincrónicos, sin embargo estos circuitos requieren un tratamiento particular, debido a que no

existen pulsos de reloj, como referencia de tiempo para controlar los cambios de estado.

En los Circuitos Secuenciales Asincrónicos las variables de entrada actúan directamente sobre el sistema, es decir que un cambio en tales variables produce un cambio sobre el estado interno. Los Circuitos Secuenciales Asincrónicos se clasifican dependiendo del tipo de entradas o del cambio en el tiempo de las estas, en dos grupos: los Circuitos Asincrónicos en Modo Fundamental y los Circuitos Asincrónicos en Modo Pulso.

### Circuitos Asincrónicos Activados por Nivel (Modo Fundamental)

Los circuitos asincrónicos operando de esta forma fueron los primeros que se implementaron en los inicios del análisis de los sistemas secuenciales en Electrónica Digital y se encuentran constituidos por un sistema combinacional, donde algunas de sus salidas se unen a las entradas formando lazos de realimentación. En la figura 150 se observa un diagrama de bloques descriptivo de este tipo de sistemas secuenciales.



**Figura 150.** Diagrama de bloques de un Circuito Asincrónico Activado por Nivel

Descripción y características de este esquema.

- La variable  $t$  representa el tiempo de retardo mínimo para que ocurra una transición y corresponde al retardo que ocurre cuando una señal viaja a través de una o más compuertas del circuito secuencial.
- En este tipo de sistemas secuenciales no se permiten cambios en forma simultánea en las variables de entrada, debido a la posible ocurrencia de estados indeterminados en las salidas.
- Se pueden presentar estados estables e inestables. Los estables son aquellos en los que el valor de estado presente es igual al estado siguiente, y los inestables son aquellos en los que el valor del estado presente es diferente al estado siguiente.
- Las variables en minúscula ( $y_n$ ) corresponden a las variables secundarias en el instante  $t$  ( $Y_t$ ), y las variables en mayúscula corresponden a las variables secundarias en el instante  $t+1$  ( $Y_{t+1}$ ).

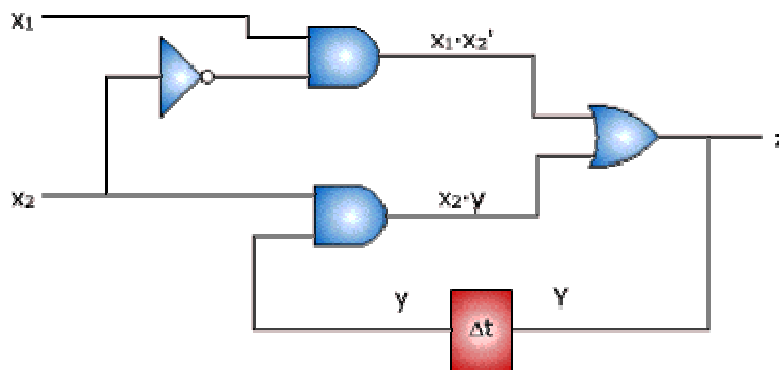
Para observar los fenómenos que pueden ocurrir en este tipo de sistemas, a continuación se describe un procedimiento para analizar los estados lógicos, el cual se desarrolla en los siguientes pasos:

1. Hallar las ecuaciones lógicas para las variables de excitación y salida del circuito.
2. Elaborar los mapas de *Karnaugh* para los estados de las variables de excitación y salida a partir de las ecuaciones halladas. Los mapas de *Karnaugh* contienen los estados secundarios *versus* los estados de salida.
3. Localizar e identificar todos los estados estables e inestables en el mapa de *Karnaugh* de las variables de excitación. Los estados estables ocurren cuando

$y_t = Y_t$ , y los estados inestables cuando  $y_t \neq Y_t$ .

4. Asignar un *nombre* (puede ser un carácter) a cada fila de la tabla.
5. Elaborar una tabla de flujo, reemplazando cada estado estable de excitación con el mismo *nombre* que tiene asignado el estado secundario, así como el de los estados inestables. Para analizar la tabla de flujo, deberán considerarse movimientos horizontales, cuando ocurran cambios en la entrada, y movimientos verticales cuando se den transiciones de estados inestables a estados estables, sin cambio en las entradas.

Para ilustrar el proceso de análisis se desarrollará un ejemplo basado en el circuito de la figura 151.



**Figura 151.** Circuito Secuencial Asíncrono de ejemplo

Este circuito tiene dos variables de entrada ( $x_1$ ,  $x_2$ ), una variable de estado interno o secundaria ( $y$ ) y una variable de salida o excitación ( $Y=Z$ ).

- *Obtención de las ecuaciones lógicas del circuito.* Según la lógica del circuito se deducen las siguientes expresiones para los estados de excitación y salida. Comparando este circuito con el de la figura 151, se observa que la variable de excitación corresponde a la variable de salida, por esta razón las expresiones son las mismas.

$$Y = x_1 \cdot x_2' + x_2 \cdot y$$

$$z = x_1 \cdot x_2' + x_2 \cdot y$$

- *Elaboración de Mapas de Karnaugh para las variables de excitación y salida.* Partiendo de las expresiones lógicas anteriores y teniendo en cuenta todas las posibles combinaciones de las variables  $x_1$ ,  $x_2$ ,  $y$  se puede llegar al mapa de *Karnaugh* de la figura 152, el cual es el mismo para  $Y$  como para  $z$ .

x1x2		00	01	11	10
y	0	0	0	1	0
	1	1	0	1	1

**Figura 152.** Mapa de *Karnaugh* para estados de excitación y salida

Esta tabla indica los cambios en el estado de la variable  $Y$  después de un cambio en las entradas  $x1$  y  $x2$ . A manera de ejemplo, observe el estado sombreado (1) en la figura 152, el cual indica que el estado actual  $Y=0$  cambia a  $Y=1$  cuando las entradas son  $x1=x2=1$ .

- *Localización de estados estables e inestables.* De la figura 152 se pueden deducir los estados estables e inestables, basta observar si los estados actuales cambian al alterar las entradas. Teniendo en cuenta lo anterior se puede concluir que los estados inestables son aquellos que están sombreados y los demás son estables, debido a que no hay cambios en el estado siguiente.
- *Asignación de nombres a cada fila de la tabla de excitación.* Las filas de la tabla sean identificadas como  $a$  y  $b$  para identificar los estados 0 y 1 de la variable  $Y$ .
- *Tabla de flujo o transición de estados lógicos.* Teniendo en cuenta que los estados de las entradas no deben tener cambios simultáneamente, en la figura 153 se muestra la tabla de flujo, donde se observa la transición de estados  $a$  y  $b$  según el estado de las entradas.

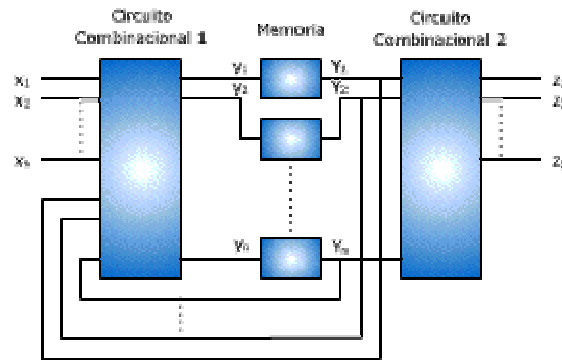
x1x2		00	01	11	10
y	a	a	a	b	a
	b	b	a	b	b

**Figura 153.** Flujo de estados



### Circuitos Asincrónicos Activados por Pulso (Modo Pulso)

Los circuitos asincrónicos operando de este modo son similares a aquellos que operan en modo fundamental, excepto que las señales de entrada corresponden a pulsos que se ocurren de forma asincrónica en la figura 154 se observa un diagrama de bloques ilustrativo sobre este tipo de sistemas.

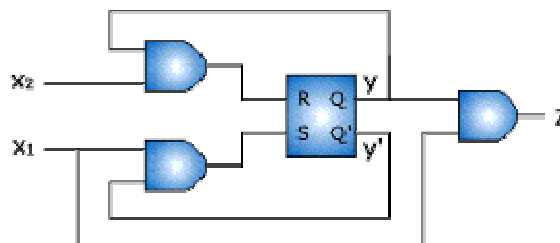


**Figura 154.** Circuito Asincrónico Activado por Pulsos

Un circuito secuencial activado por pulsos, se caracteriza por cumplir las siguientes condiciones:

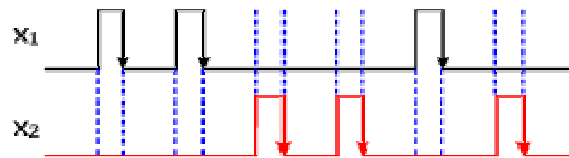
- Como mínimo, una de las entradas debe ser un pulso.
- Los cambios en los estados internos ocurren únicamente por la presencia de un pulso en las terminales de entrada.
- Cada estado de entrada, desencadena únicamente un cambio en el estado interno del circuito.
- No se permiten dos o más pulsos en forma simultánea en las señales de entrada. En caso de incumplirse esta condición la única forma de analizar el circuito es con un diagrama de tiempos.
- Existen dos tipos de circuitos en esta modalidad de funcionamiento: La máquina de estados de Mealy y Moore (Ver Teoría de máquinas de estado (FSM))

Para entender el funcionamiento de este tipo de circuitos, se desarrollará un ejemplo con base en el circuito de la figura 155.



**Figura 155.** Circuito Asincrónico de Ejemplo

Para comenzar el análisis considere que los pulsos de entrada ocurren en la secuencia que se observa en la figura 156. Note que los estados de las entradas son complementarios y las transiciones ocurren en instantes de tiempo diferentes, lo cual es una característica particular de las entradas de estos sistemas secuenciales.

Figura 156. Secuencia de pulsos para  $x_1$  y  $x_2$ 

Para analizar el estado de las variables del circuito se deben deducir las expresiones lógicas para S, R y z. De la figura 155, se tiene:

$$S = x_1 \cdot y'$$

$$R = x_2 \cdot y$$

$$z = x_1 \cdot y$$

A partir de las expresiones lógicas se puede construir el diagrama de tiempos para las variables del circuito. En la figura 157 se observan las transiciones de los estados correspondientes a la secuencia de las señales de entrada.

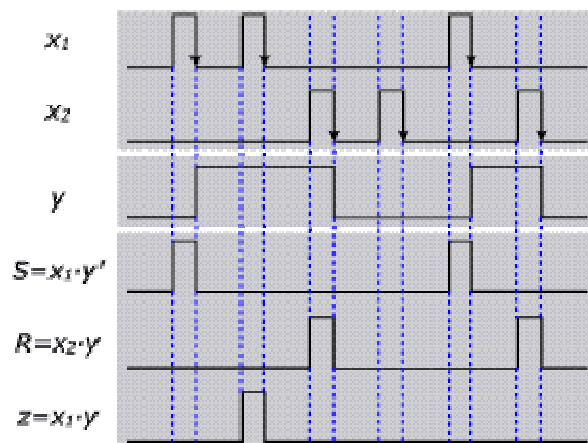


Figura 157. Diagrama de tiempo del circuito de la figura 7.3.6

La figura 158 muestra los estados siguientes y los estados de salida de la forma "estado siguiente/estado salida" (y/z).

		$x_1 x_2$		
		00	01	10
y	0	0/0	0/0	1/0
	1	1/0	0/0	1/1

Figura 158. Estados de Excitación y Salida

En la tabla no se tuvo en cuenta la columna correspondiente a la entrada  $x_1 x_2 = 11$ , debido a que los circuitos secuenciales asíncronos no admiten entradas activas de forma simultánea. Esta tabla de estados se puede simplificar aun más debido a que el

estado 00 no implica ningún cambio en los estados del circuito, así que la columna correspondiente se puede suprimir, sin alterar el análisis. Teniendo presente esta condición, la figura 158 se reduce a la figura 159. Observe que los estados de las entradas son complementarios, lo cual es característico de una señal pulsada.

x1x2		01	10
y			
0		0/0	1/0
1		0/0	1/1

**Figura 159.** Estados de Excitación y Salida

## 6.12. Ejemplos de Control Secuencial

Los sistemas combinacionales y secuenciales tienen gran variedad de aplicaciones en la vida real. En la mayoría de sistemas digitales encontrados en la práctica se incluyen elementos que memorizan la información, por lo cual se requieren de circuitos secuenciales.

El objetivo de esta lección consiste en dar aplicabilidad a la teoría vista en este capítulo, mediante dos ejemplos sencillos, con los cuales se harán uso de las herramientas de análisis y diseño de circuitos secuenciales: la implementación de un semáforo y un control de un motor de pasos.

### Implementación de un Semáforo:

Construir el circuito lógico para un semáforo que responda a la siguiente secuencia: Verde, Amarillo, Rojo y Rojo/Amarillo.

El semáforo tiene cuatro estados, los cuales se pueden representar con 2 *FLIP-FLOPs*, sin embargo para asignar el tiempo de duración de cada estado se emplearán 3 *FLIP-FLOPs*, de los cuales se pueden obtener 8 estados, cuyos tiempos se pueden distribuir de la siguiente forma:

- Verde (3 ciclos)
- Amarillo (1 ciclo)
- Rojo (3 ciclos)
- Rojo-Amarillo (1 ciclo)

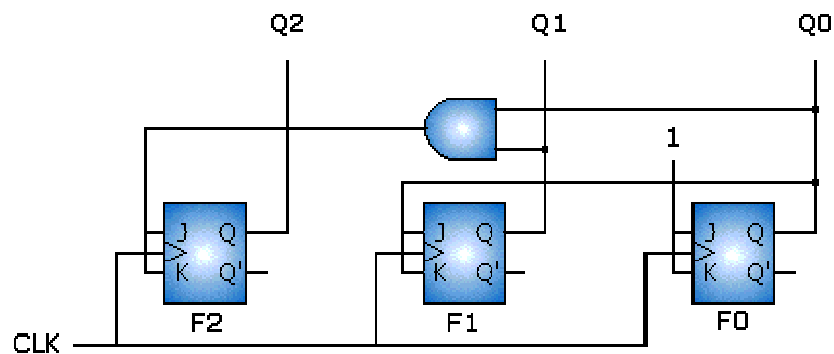
Donde cada ciclo representa una transición en la señal de reloj. Observe que la duración de la secuencia de los cuatro estados es de 8 ciclos.

El primer paso para realizar el diseño consiste en asignar los estados lógicos, como se puede notar en la tabla 69. Esta asignación de estados se puede hacer de forma libre y no necesariamente debe corresponder a una secuencia binaria, sin embargo, en este caso por comodidad sean establecido de esta forma para implementar el circuito con base en un contador sincrónico de tres *bits*.

Color	Salidas de los FLIP-FLOPs			Salidas al Semáforo		
	$Q_2$	$Q_1$	$Q_0$	V	A	R
Verde	0	0	0	1	0	0
	0	0	1	1	0	0
	0	1	0	1	0	0
Amarillo	0	1	1	0	1	0
Rojo	1	0	0	0	0	1
	1	0	1	0	0	1
	1	1	0	0	0	1
Rojo-Amarillo	1	1	1	0	1	1

**Tabla 69.** Asignación de estados

En la figura 160 se observa un contador síncrono de tres *bits* construido con *FLIP-FLOPs JK*, a partir del cual se realizará el diseño. El objetivo de hacer uso del contador es emplear sus salidas ( $Q_2$ ,  $Q_1$  y  $Q_0$ ) para generar los estados de las variables V, A y R (Verde, Amarillo y Rojo) del semáforo.



**Figura 160.** Contador de tres bits

El siguiente paso consiste en deducir la lógica combinacional adicional para generar los estados de las variables V, A y R. Para ello se deben construir los mapas de *Karnaugh* y obtener las ecuaciones lógicas. En la figura 161 se muestran los mapas con las ecuaciones resultantes para cada variable.

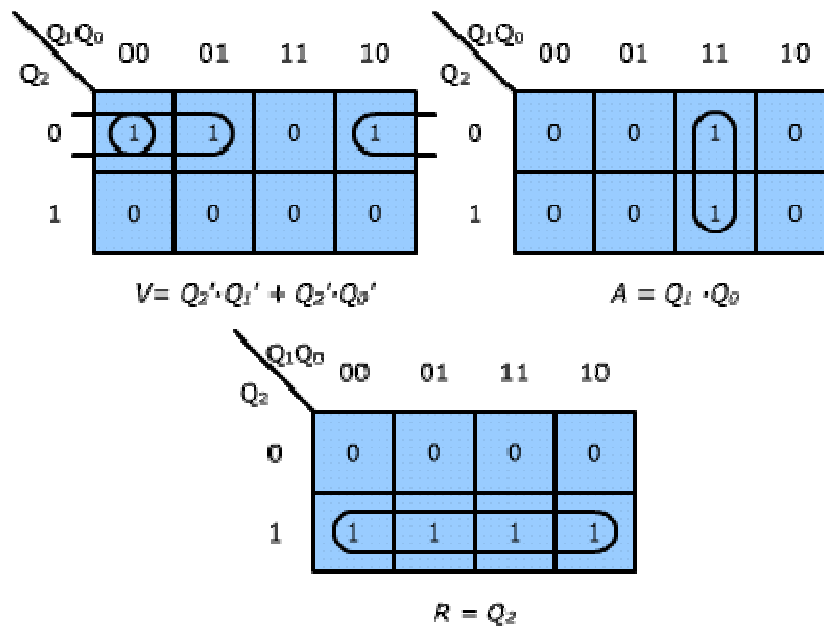


Figura 161. Mapas de Karnaugh

### Motor paso a paso operando en forma unipolar

Un motor de pasos es un tipo especial de motor diseñado para rotar un determinado ángulo como respuesta a una señal en su circuito de control. Estos motores se utilizan en varios sistemas de control de posición debido a la precisión que manejan.

Este tipo de motor puede tener una o dos bobinas por fase. Los que tienen una bobina por fase se conocen como motores de tres hilos y los que tienen dos bobinas por fase se conocen como motores de devanado partido. Para este ejemplo se empleará un motor de fase partida, como el que se indica en la figura 162. Observe la forma en que debe ser conectado para hacer el control.

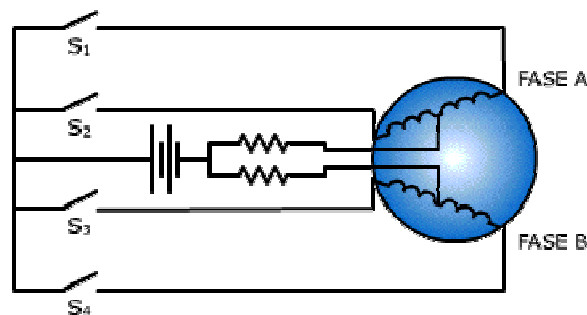


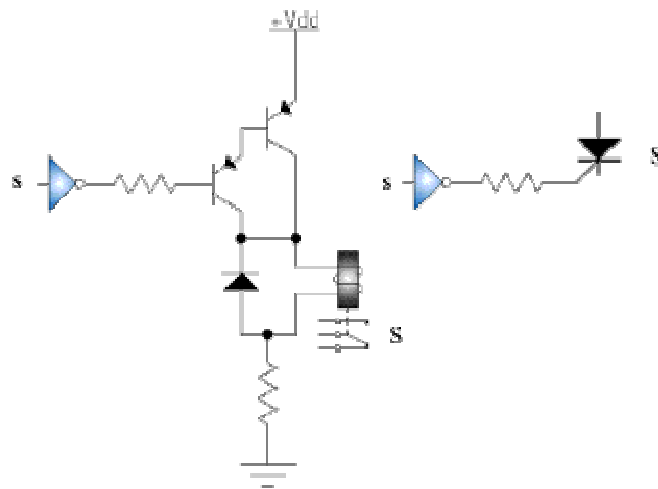
Figura 162. Motor de pasos de devanado partido

En este ejemplo se hará el diseño del circuito de control para manejar cuatro pasos, los cuales corresponden a la posición de los interruptores se indican en la tabla 70.

Numero de paso	Estado de los interruptores			
	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>
1	ON	OFF	OFF	ON
2	ON	OFF	ON	OFF
3	OFF	ON	ON	OFF
4	OFF	ON	OFF	ON

**Tabla 70.** Secuencia de estados de los interruptores (4 pasos)

Los interruptores se pueden controlar de dos formas, ya sea con tiristores (SCR's) o mediante el uso de relevos. En la figura 163 se observan las dos opciones para manejar los interruptores.



**Figura 163.** Interruptor por relevo y de estado solido

Observando la tabla 70, se puede notar que los estados de los interruptores S<sub>1</sub> y S<sub>2</sub>, son complementarios, al igual que los interruptores S<sub>3</sub> y S<sub>4</sub>, lo cual simplifica el diseño del circuito.

El primer paso para realizar el diseño de la unidad de control, consiste en asignar los estados lógicos y seleccionar el tipo de *FLIP-FLOP* con el cual se implementará el circuito lógico.

En la tabla 71 se relacionan los estados lógicos de las salidas y los estados de las entradas *j* y *k* de los *FLIP-FLOPs*.

Note que las variables S<sub>2</sub> y S<sub>4</sub> no se tuvieron en cuenta, debido a que sus estados son el complemento de S<sub>1</sub> y S<sub>3</sub> respectivamente.

Estado Actual		Entrada	Estado Siguiente		Estados de los FLIP-FLOPs			
S1	S3		S1	S3	J1	K1	J3	K3
1	0	0	0	0	X	1	0	X
1	1	0	1	0	X	0	X	1
0	1	0	1	1	1	X	X	0
0	0	0	0	1	0	X	1	X
1	0	1	1	1	X	0	1	X
1	1	1	0	1	X	1	X	0
0	1	1	0	0	0	X	X	1
0	0	1	1	0	1	X	0	X

Tabla 71. Tabla de estado

El siguiente paso consiste en construir los mapas de *Karnaugh* para los estados de los *FLIP-FLOPs* ( $J_1$ ,  $K_1$ ,  $J_3$ ,  $K_3$ ). Tales estados se indican en los mapas de *Karnaugh* mostrados en la figura 164 con las ecuaciones lógicas correspondientes.

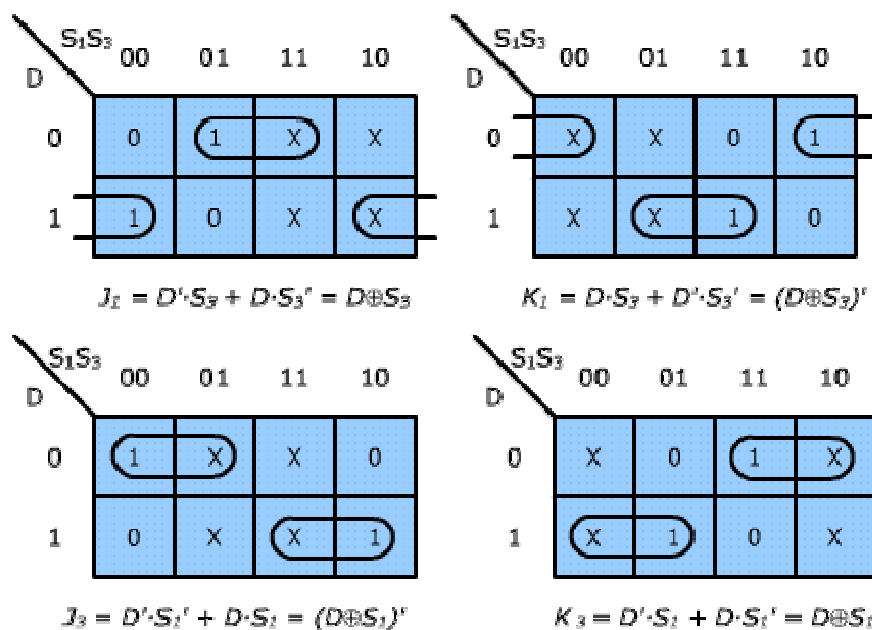
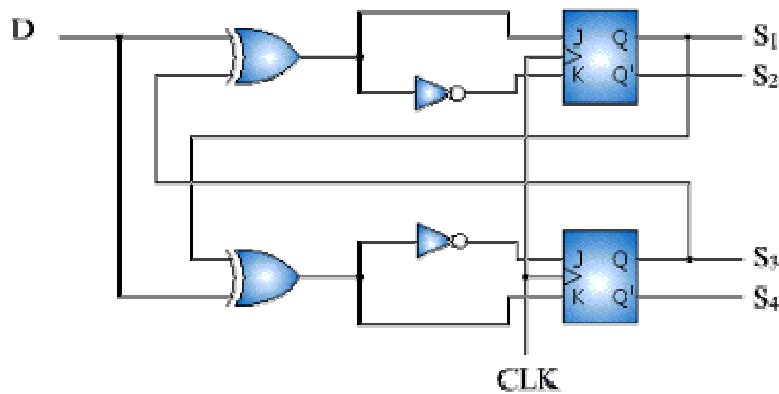


Figura 164. Mapas de Karnaugh

El último paso del diseño consiste en construir el circuito lógico a partir de las ecuaciones lógicas obtenidas, el cual se muestra en la figura 165.



**Figura 165.** Diseño final del circuito lógico

Observe que los estados S2 y S4 no se tuvieron en cuenta en el diseño debido a que los FLIP-FLOPs por defecto entregan en sus salidas una variable y su complemento.

## 7. Dispositivos Lógicos Programables

Una forma rápida y directa de integrar aplicaciones se logra con la lógica programable, la cual permite independizar el proceso de fabricación del proceso de diseño fuera de la fábrica de semiconductores. El sistema desplaza los errores de alambrado al campo exclusivo de la programación. Los sistemas con estas características se pueden borrar y reprogramar en casos de cambios o revisiones. El resultado es la reducción del espacio físico de la aplicación. El diseño está basado en bibliotecas y mecanismos específicos de mapeado de funciones.

La mayor parte de los diseños de nivel de sistema incorporan diversos dispositivos, como son las memorias *RAM*, *ROM*, controladores, procesadores, etc., que se interconectan mediante gran cantidad de dispositivos lógicos de propósito general, frecuentemente denominados lógica de unión ("*glue logic*"). En los últimos años, los dispositivos *PLD* (*Programmable Logic Device*) han comenzado a reemplazar muchos de los antiguos dispositivos de unión, *SSI* y *MSI*.

El uso de dispositivos *PLD* proporciona una reducción en el número de circuitos integrados. Por ejemplo, en los sistemas de memoria de las computadoras, los *PLD* pueden utilizarse para decodificar direcciones de memoria y generar señales de escritura en memoria.

En muchas aplicaciones, los *PLD* y, en concreto, las matrices lógicas programables (*PAL*, *Programmable Array Logic*) y las matrices lógicas genéricas (*GAL*, *Generic Array Logic*) pueden emplearse para reemplazar dispositivos lógicos *SSI* y *MSI*, consiguiendo con ello una reducción de etapas y de los costos.

Por las razones anteriores el diseño lógico hoy día se realiza con *PLDs*. Un *PLD* está formado por una matriz de puertas *AND* y puertas *OR*, que se pueden programar para conseguir funciones lógicas específicas.

El diseño con *PLDs* señala las siguientes ventajas en relación a la lógica cableada:

- Economía.
- Menos espacio en los impresos.
- Se mantiene la reserva del diseño.



- Se requiere tener menos inventarios que con circuitos estándar *SSI*, *MSI*.
- Menos alambrado.

**7.1. Tipos de PLD:** Los *PLD* se dividen en dos clases:

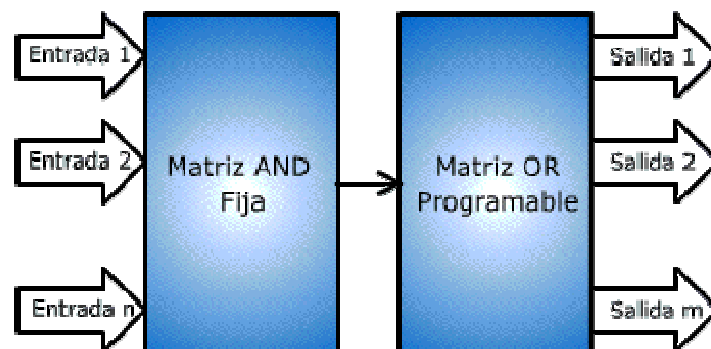
1. *PLDs* combinatorios: Constituidos por arreglos de compuertas *AND* – *OR*. El usuario define las interconexiones y en esto consiste la programación.
2. *PLDs* secuenciales: Además de los arreglos de compuertas, incluyen *FLIP* – *FLOPs* para programar funciones secuenciales como contadores y máquinas de estado

## **7.2. Estructura de los Dispositivos Lógicos Programables Básicos**

Los *PLD* se clasifican de acuerdo con su estructura, la cual es básicamente la ordenación funcional de los elementos internos que proporciona al dispositivo sus características de operación específicas.

### **Memoria programable de sólo lectura PROM (PROM, Programmable Read Only Memory)**

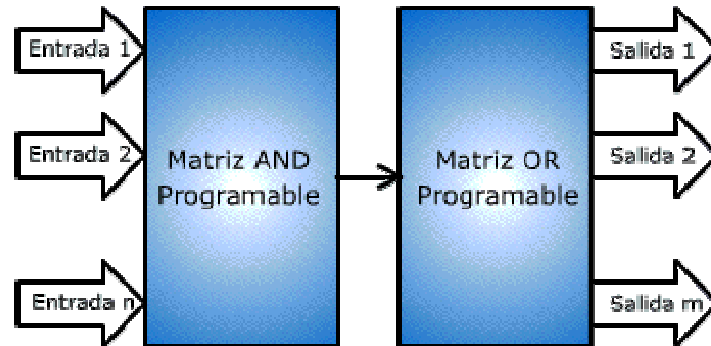
La *PROM* está formada por un conjunto fijo (no programable) de puertas *AND* conectadas como decodificador y una matriz programable *OR*. La *PROM* se utiliza como una memoria direccionable y no como un dispositivo lógico (Ver Figura 166).



**Figura 166.** Diagrama de bloques de una PROM (Programmable Read-Only Memory).

### Arreglo Lógico Programable PLA (PLA, Programmable Logic Array)

El *PLA* es un *PLD* formado por una matriz *AND* programable y una matriz *OR* programable. La *PLA* ha sido desarrollada para superar algunas de las limitaciones de las memorias *PROM* (Ver Figura 167).



**Figura 167.** Diagrama de bloques de una PLA (Programmable Logic Array).

En la actualidad existen soluciones con Dispositivos Lógicos programables complejos que combinan arquitectura superior y software de gran alcance, ofreciendo un nivel sin precedente en la flexibilidad del diseño.

### 7.3. Herramientas para la Automatización del Diseño Electrónico (EDA Tools)

Las herramientas *EDA* ("*Electronic Design Automation*") son las herramientas de *hardware* y *software* utilizadas en el diseño de sistemas electrónicos.

El diseño de *hardware* tiene un inconveniente que no existe en el desarrollo de *software*. El problema es el alto costo en el ciclo de diseño, desarrollo del prototipo, pruebas y reinicio del ciclo. La etapa de costo más elevado es el prototipo. Por necesidad del mercado, se impone la reducción de costos en esta etapa, con el fin de incluir la fase de desarrollo del prototipo al final del proceso, evitando la repetición de varios prototipos, razón por la cual se encarece el ciclo. La introducción de la fase de simulación y verificación de circuitos utilizando herramientas *EDA*, hace no necesaria la comprobación del funcionamiento del circuito por medio de la implementación física del prototipo.

Las herramientas *EDA* están presentes en todas las fases del ciclo de diseño de circuitos. Primero en la fase de generación del sistema que puede representarse en un diagrama esquemático, en bloques o de flujo.

Se encuentra también la fase de simulación y comprobación de circuitos, donde diferentes herramientas permiten verificar el funcionamiento del sistema. Estas simulaciones pueden ser de eventos, funcionales, digitales o eléctricas, de acuerdo al nivel requerido. Después están las herramientas *EDA* utilizadas en la síntesis y programación de circuitos digitales en dispositivos lógicos programables. Existen, además, las herramientas *EDA* orientadas a la fabricación de circuitos. En el caso del diseño de *hardware* estas herramientas sirven para la realización de *PCBs* ("*Printed Circuit Boards*" o placas de circuito impreso), o para desarrollar circuitos integrados de aplicación específica como *ASICs* ("*Application Specific Integrated Circuits*").

Las principales características y finalidad de algunas herramientas *EDA* que intervienen en el diseño de circuitos son:

1. Lenguajes de Descripción de Circuitos.
2. Diagramas Esquemáticos.
3. Grafos y Diagramas de Flujo.
4. Simulación de Eventos.
5. Simulación Funcional.
6. Simulación Digital.
7. Simulación Eléctrica.
8. Diseño de *PCBs*.
9. Diseño de Circuitos Integrados.
10. Diseño con Dispositivos Programables.

Para la automatización del diseño electrónico se utilizan herramientas *EDA*.

### **Ventajas de la metodología de diseño que usa herramientas EDA**

Entre las ventajas de la metodología de diseño con el empleo de herramientas *EDA* está la reducción del diseño, la posibilidad de dividir un proyecto en módulos que se desarrollan por separado, la independencia del diseño con respecto a la tecnología, la posibilidad de la reutilización de los diseños, la optimización de los circuitos y las simulaciones posibles con las herramientas.

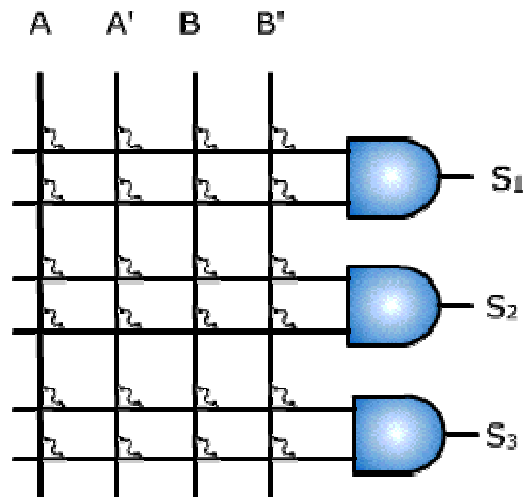
Con la aparición de herramientas *EDA* cada vez más complejas, que integran en el mismo marco de trabajo las herramientas de descripción, síntesis, simulación y realización; apareció la necesidad de disponer de un método de descripción de circuitos que permitiera el intercambio de información entre las diferentes herramientas que componen el ciclo de diseño. En principio se utilizó un lenguaje de descripción que permitía, mediante sentencias simples, describir completamente un circuito. A estos lenguajes se les llamó *Netlist* puesto que eran simplemente eso, un conjunto de instrucciones que indicaban las interconexiones entre los componentes de un diseño.

### **7.4. Principios y Aplicaciones de los Dispositivos Lógicos Programables como las PALs y las GALs.**

Una matriz programable es una red de conductores distribuidos en filas y columnas con un fusible en cada punto de intersección. Las matrices pueden ser fijas o programables. Todos los *PLD* están formados por matrices programables.

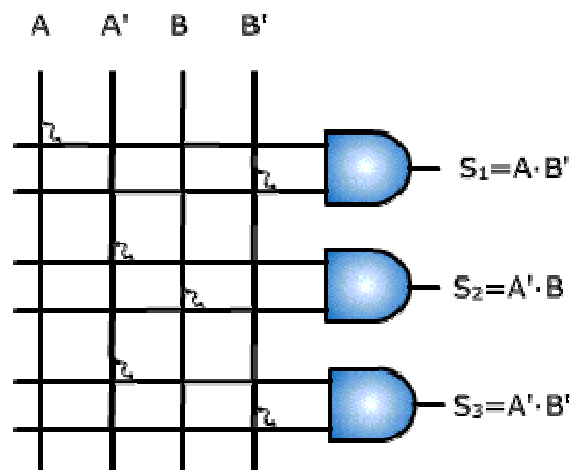
**Estructura Interna de un PLD:** Está formada por un arreglo de compuertas *AND* y *OR* interconectadas a través de fusibles.

**Matriz AND:** está formada por una red de compuertas *AND* conectadas a través conductores y fusibles en cada punto de intersección. Cada punto de intersección entre una fila y una columna se denomina celda. La figura 168 muestra un arreglo de compuertas no programado.



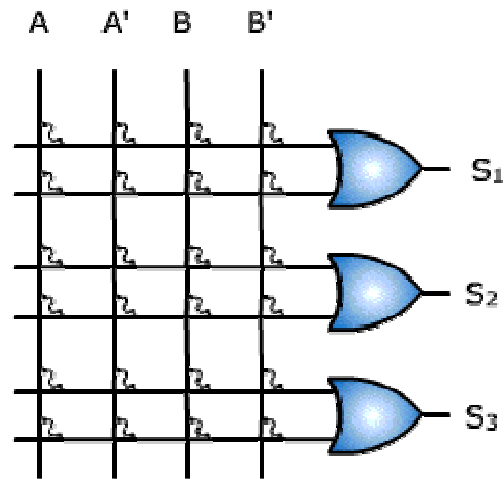
**Figura 168.** Arreglo AND No Programado.

Cuando se requiere una conexión entre una fila y una columna, el fusible queda intacto y en caso de no requerirse la conexión, el fusible se abre en el proceso de programación. La figura muestra 169 un arreglo *AND* programado.



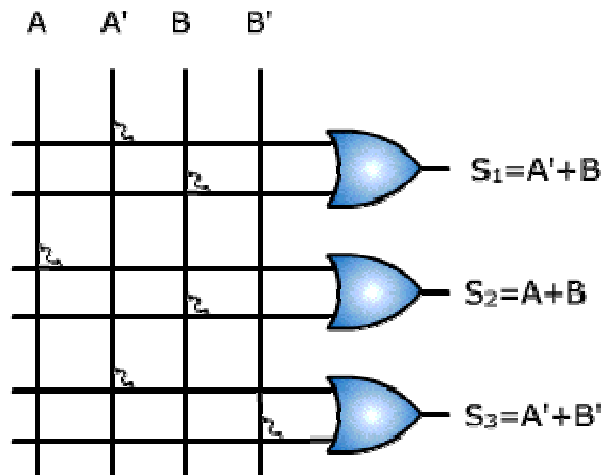
**Figura 169.** Arreglo AND Programado.

**Matriz OR:** está formada por una red de compuertas *OR* conectadas a través conductores y fusibles en cada punto de intersección. La figura 170 muestra un arreglo de compuertas no programado.



**Figura 170.** Arreglo OR No Programado.

La figura 171 muestra un arreglo OR programado.

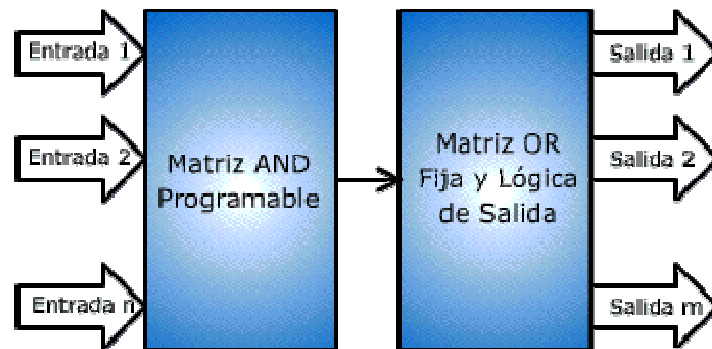


**Figura 171.** Arreglo OR Programado.

Los dispositivos lógicos programables que se usan más comúnmente para la implementación lógica son la *PAL* y la *GAL*.

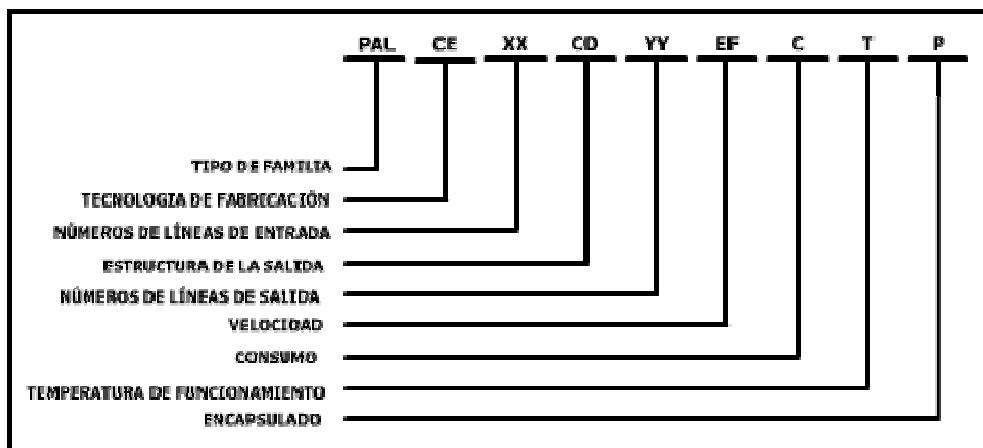
### Lógica de Arreglos Programables (*PAL*, Programmable Array Logic)

La *PAL* es un *PLD* que se ha desarrollado para superar ciertas desventajas de la *PLA*, tales como los largos retardos debidos a los fusibles adicionales que resultan de la utilización de dos matrices programables y la mayor complejidad del circuito. La *PAL* básica está formada por una matriz *AND* programable y una matriz *OR* fija con la lógica de salida (Ver figura 172). Esta estructura permite implementar cualquier suma de productos lógica con un número de variables definido, sabiendo que cualquier función lógica puede expresarse como suma de productos. La *PAL* se implementa con tecnología bipolar (*TTL* o *ECL*).



**Figura 172.** Diagrama de bloques de una PAL (Programmable Logic Array)

**Nomenclatura de una PAL:** Los líderes en fabricación de *PLDs*, *Texas Instruments* y *AMD*, tienen una notación para identificar los dispositivos. Por ejemplo, la estructura en *PLD AMD* es:



**Figura 173.** Diagrama de bloques de una PAL (Programmable Logic Array)

Dentro de la estructura de salida se tienen las posibilidades contenidas en la tabla 72.

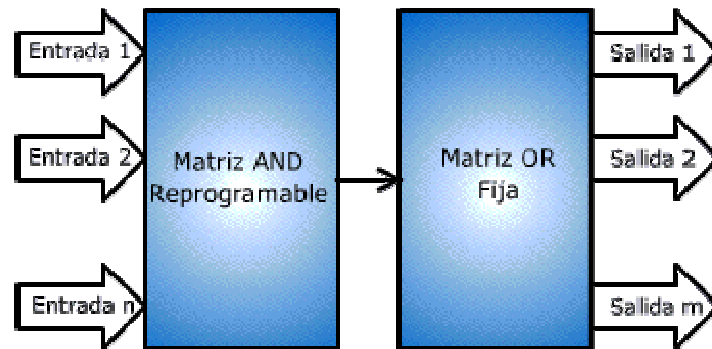
Códigos	Tipos de Salidas
L	Combinatoria con nivel bajo activo.
H	Combinatoria con nivel alto activo.
R	Registro.
RA	Registro asíncrono.
X	Registro O exclusivo.
V	Vesátil.
M	Macro célula.

**Tabla 72.** Tipos de Salidas de una PAL.

**PALs comerciales:** En el mercado se manejan referencias como la *PAL16L8*, *PAL20L8*, *PAL20V8* y *PAL20X8*.

### Matriz Lógica Genérica (GAL, Generic Array Logic)

La GAL se forma con una matriz AND reprogramable y una matriz OR fija, con una salida lógica programable. La figura 174 muestra el diagrama de bloques de una GAL. Esta estructura permite implementar cualquier expresión lógica suma de productos con un número de variables limitado.

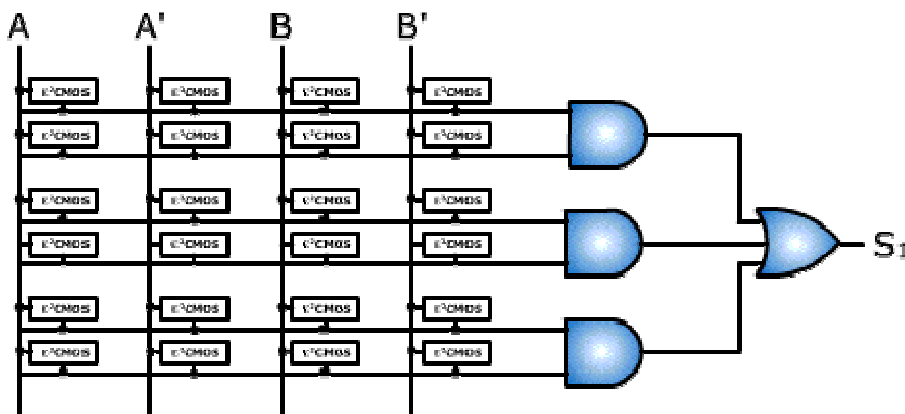


**Figura 174.** Diagrama de Bloques de una GAL (**Generic Array Logic**).

Las dos principales diferencias entre los dispositivos GAL y PAL son:

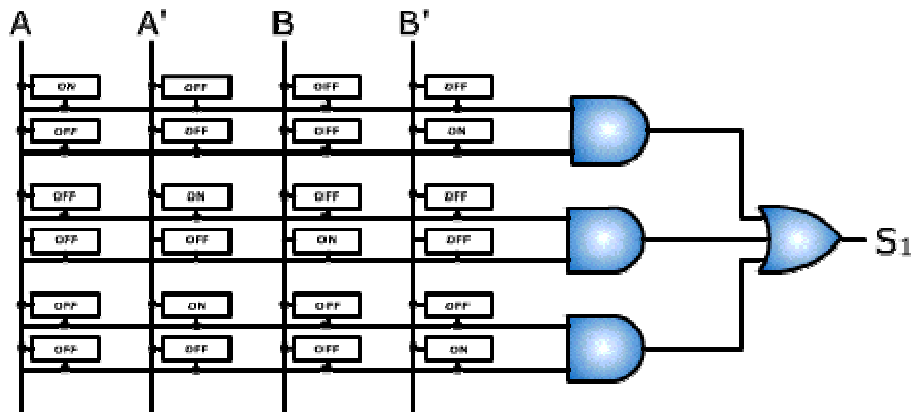
- a) La GAL es reprogramable y
- b) La GAL tiene configuraciones de salida programables. La GAL se puede programar una y otra vez, ya que usa tecnología ECMOS (*Electrically Erasable CMOS, CMOS borrrable eléctricamente*).

En la figura 175 se ilustra la estructura básica de una GAL con dos variables de entrada y una de salida. La matriz reprogramable es esencialmente una red de conductores ordenados en filas y columnas, con una celda CMOS eléctricamente borrrable ( $E^2$ CMOS) en cada punto de intersección, en lugar de un fusible como en el caso de las PAL. Estos PLDs son borrrables y reprogramables. El transistor CMOS tiene 2 compuertas, una de ellas totalmente aislada, flotante. Para programar cada celda se aplica o no una tensión mayor a  $V_{DD}$  (*alta*) en la compuerta no flotante. Al aplicar esta tensión el dieléctrico conduce y la compuerta flotante se carga negativamente, dejando en operación normal siempre abierto el transistor.



**Figura 175.** Estructura Básica de una GAL (**Generic Array Logic**)

En la figura 176 se muestra un ejemplo de una sencilla matriz GAL programada para obtener la suma de tres productos.



**Figura 176.** Programación de una GAL (**Generic Array Logic**).

El borrado se puede hacer de dos formas:

- Con luz ultravioleta (*UV*): exponiendo el transistor de 5 a 20 minutos a luz *UV*, el dieléctrico conduce y permite la descarga de la compuerta flotante. Para este borrado el chip lleva una ventana de cuarzo transparente.
- Borrado eléctrico: Es el más usado hoy en día. La capa que aísla la compuerta flotante es más delgada. Al aplicar una tensión alta con polaridad contraria, la compuerta flotante se descarga porque el dieléctrico conduce. Las ventajas más importantes de esta técnica son una descarga rápida, no se requiere *UV* y no se requiere sacar el chip de su base.

**GALs comerciales:** Las diversas GAL tienen el mismo tipo de matriz programable. Se diferencian en el tamaño de la matriz, en el tipo de *OLMC* (Las macroceldas Lógicas de Salida que contienen circuitos lógicos programables que se pueden configurar como entrada o salida combinacional y secuencial) y en los parámetros de funcionamiento, tales como velocidad y disipación de potencia.

Referencia	Número de Pines	t <sub>PD</sub>	I <sub>CC</sub> (mA)	Características
GAL16V8A	20	10, 15, 25	55, 115	E <sup>2</sup> CMOS PLD Genérica
GAL18V10	20	15, 20	115	E <sup>2</sup> CMOS PLD Universal
GAL22V8A	24	10, 15, 25	55, 115	E <sup>2</sup> CMOS PLD Genérica
GAL22RA10	24	15, 20	115	E <sup>2</sup> CMOS PLD Universal
GAL22V10	24	10, 15, 25	130	E <sup>2</sup> CMOS PLD Universal
GAL26CV12	28	15, 20	130	E <sup>2</sup> CMOS PLD Universal
GAL6001	24	30, 35	150	E <sup>2</sup> CMOS FPLA
ispGAL16Z8	24	30, 35	190	E <sup>2</sup> CMOS PLD Programable en Circuito

**Tabla 73.** Familias GAL del fabricante Lattice

En la mayoría de las aplicaciones en electrónica digital se requiere del almacenamiento de información de forma temporal para efectuar operaciones lógicas. Debido a que los *FLIP-FLOPs* tienen esta característica, la gran parte de los



**Dispositivos Lógicos Programables “PLDs”** existentes en el mercado tienen incorporados estos dispositivos en su estructura interna.

Estos dispositivos lógicos se conocen como *PLDs* de registro y son empleados para construir máquinas de estado de propósito especial, además de la ventaja que ofrecen para reducir el tamaño de los circuitos. Se puede agregar que los *PLDs* facilitan el ruteado de las placas de circuito impreso debido a la libertad de asignación que proporcionan, y además permiten realizar modificaciones posteriores del diseño.

Los *PLDs* secuenciales se componen de un arreglo programable de compuertas *AND* seguido de un arreglo fijo de compuertas *OR*, dispuestos de la misma manera que en los *PLDs* combinatorios. Su diferencia con los *PLDs* combinatorios se debe a que el estado de las salidas se almacena en *FLIP-FLOP*s cuando se presenta un flanco activo en la señal de reloj del *PLD*.

### 7.5. Arquitectura de Diversos PLD's Secuenciales

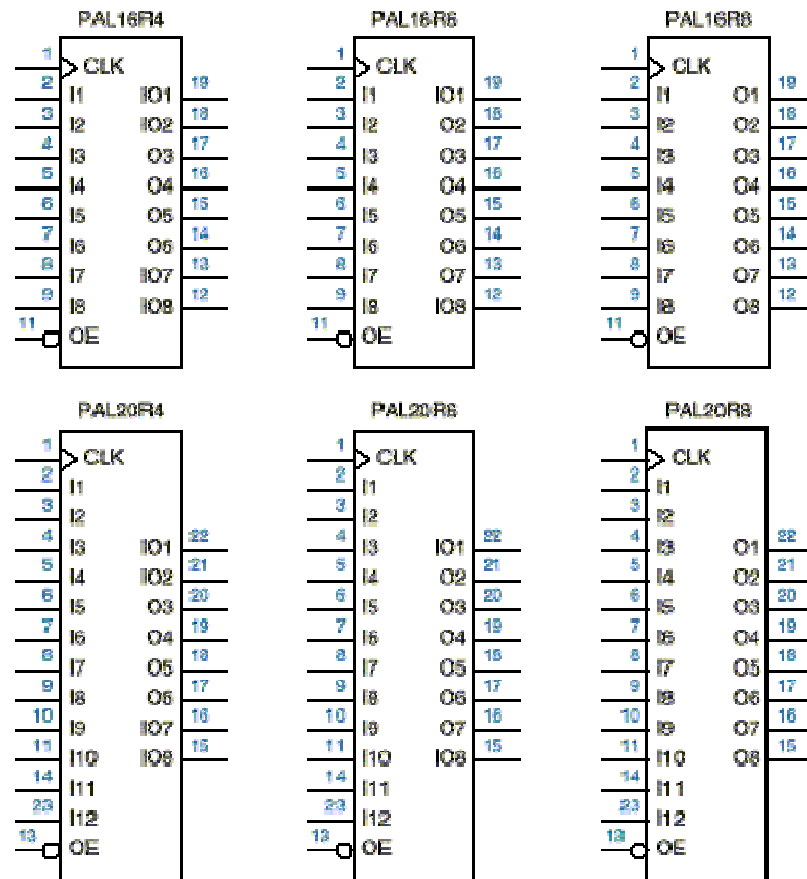
Los *PLDs* secuenciales se encuentran clasificados en dos tipos: "no reprogramables" y "reprogramables". A continuación veremos estas dos categorías de *PLDs* y sus diferencias.

***PLDs no reprogramables:*** En este tipo de *PLD's* es posible programar el arreglo de compuertas sólo una vez, de tal manera que no es posible hacer modificaciones posteriores al estado de los fusibles, quedando únicamente operando con la lógica definida por las conexiones internas programadas. Estos dispositivos son conocidos por la sigla *PAL* (*Programmable Array Logic*).

En la tabla 74 se observa una lista de algunos *PLDs* secuenciales de uso general. Los *PLDs PAL16XX* indicados en la tabla tiene el mismo arreglo de compuertas de 16 variables. Los *PLDs* de la familia *PAL20XX* tienen un arreglo de compuertas similar con 20 variables. En la figura 177 se observan los esquemas lógicos de los *PLDs* relacionados en la tabla 74.

PLD	Nº de pines	Entradas por compuerta AND	Entradas principales	Salidas combinacionales bidireccionales	Salidas tipo registro	Salidas combinacionales
PAL16R4	20	16	8	4	4	0
PAL16R6	20	16	8	2	6	0
PAL16R8	20	16	8	0	8	0
PAL20R4	24	20	12	4	4	0
PAL20R6	24	20	12	2	6	0
PAL20R8	24	20	12	0	8	0

**Tabla 74.** Descripción de *PLDs* secuenciales no reprogramables de uso general



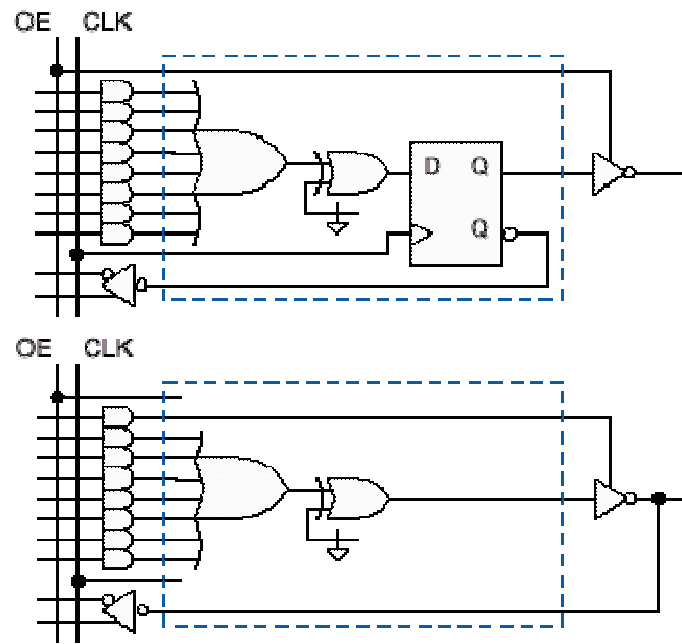
**Figura 177.** Diagramas lógicos de PLDs secuenciales no reprogramables (Copyright © 1999 by John F. Wakerly)

**PLDs reprogramables:** Estos PLDs utilizan tecnología *EEPROM* (*Electrical Erasable Programmable ROM*) y se conocen con el nombre de *GALs* (*Generic Array Logic*). Estos dispositivos a diferencia de los anteriores permiten modificar la disposición interna de las conexiones de las compuertas después de haber sido programados.

Este dispositivo tiene 20 pines distribuidos de la siguiente forma:

- 8 entradas dedicadas (pines 2 a 9).
- 8 salidas de registro programables (pines 12 a 19).
- 1 entrada de reloj (pin 1).
- 1 entrada de habilitación (pin 11).
- 2 entradas de alimentación (pines 10 y 20).

Las salidas se pueden programar como salida secuencial o como salida combinacional dependiendo del estado de los fusibles de selección ubicados en la *macroelda lógica* de cada salida. La *macroelda* corresponde al conjunto de elementos agrupados en cada salida, incluyendo la compuerta OR).



**Figura 178.** Macroceldas lógicas para el PLD GAL16V8. (Copyright © 1999 by John F. Wakerly).

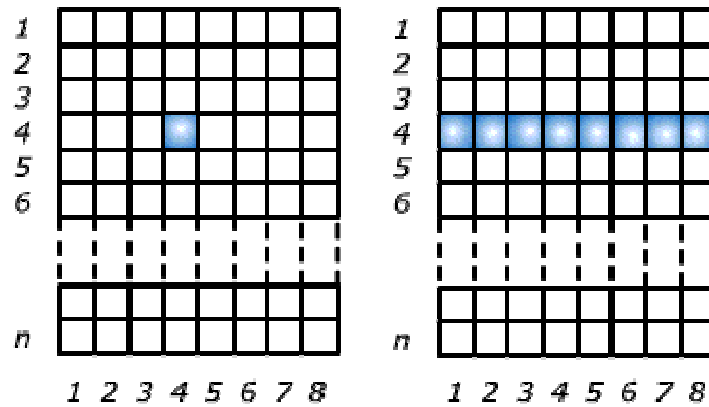
Estas celdas son conocidas como *OLMCs* de la sigla en inglés *Output Logic MacroCell* y en la figura 178 se observa la estructura interna de una de estas celdas en sus dos configuraciones disponibles (salida secuencial y salida combinacional).

**7.6. Memorias:** La mayoría de los procesos lógicos en electrónica digital se encuentran constituidos por sistemas que manipulan la información binaria para dar como resultado una o varias salidas.

En el proceso de manipular la información, los sistemas requieren del almacenamiento temporal o permanente de los estados lógicos. Un ejemplo de este tipo de sistemas son los microcomputadores, los cuales necesitan del almacenamiento tanto de datos como de los programas que manipulan la información.

### **Aspectos Generales sobre Memorias**

Las unidades de memoria son módulos conformados por un conjunto de cerrojos o condensadores agrupados de tal forma que almacenan varias palabras binarias de  $n$  bits. Cada una de ellas tiene la capacidad de almacenar un bit de información ( $1$  o  $0$ ), y se conocen con el nombre de celdas de memoria. Las celdas o *bits* de memoria se ubican mediante la fila y la columna en la que se encuentra. En la figura 179 se observa como ubicar un *bit* y una palabra dentro de una memoria.



**Figura 179.** Ubicación de la información en una memoria

Las palabras binarias se identifican con una dirección la cual define la ubicación dentro del arreglo y generalmente se designa con un número binario, octal o hexadecimal. En la mayoría de las aplicaciones se asocian en grupos de ocho unidades para formar *bytes* y el tamaño de las palabras en las memorias actuales está entre 4 y 64 *bits*.

El parámetro básico de una memoria es su capacidad, la cual corresponde al total de unidades que puede almacenar. Como ejemplo, la memoria de la figura 179 tiene una capacidad de  $8n$  bits, que en otras palabras representa  $n$  *bytes*.

Regularmente estas memorias en la actualidad se consiguen en tamaños del orden *megabytes*.

El tiempo de acceso es otro parámetro importante en las memorias. Este corresponde al tiempo que tarda la memoria en acceder a la información almacenada en una dirección.

Generalmente este tiempo se designan como  $t_{acc}$  en las fichas técnicas de estos dispositivos. En tabla 75 se indican los tiempos de acceso de las memorias en Circuito Integrado comparados con los tiempos de otros tipos de memoria.

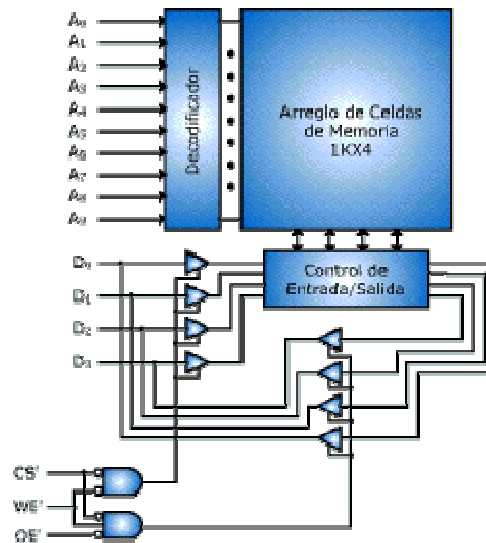
Memoria	Tiempo de Acceso
Núcleo de Ferrita	0.3 - 1.0 us
Cinta Magnética	5 ms - 1s
Disco Magnético	10ms - 50 ms
CD ROM	200 ms – 400 ms
Memorias Integradas MOS	2ns – 300 ns
Memorias Integradas Bipolares	0.5ns – 30 ns

**Tabla 75.** Comparación de tiempos de acceso de diversos tipos de memorias

### Operaciones básicas de una Memoria

La función básica de las memorias es almacenar información. Sin embargo las memorias tienen la función específica de escribir y leer los datos en su interior. En la

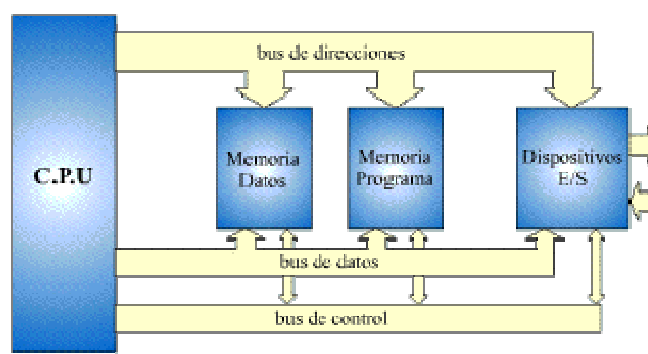
figura 180 se observa la estructura básica de una memoria de 1K de 4 bits, en la cual se indican sus partes básicas.



**Figura 180.** Esquema descriptivo de una Memoria

En la figura 180 la entrada de direcciones ( $A_0$  a  $A_9$ ), como su nombre lo indica, define la posición a escribir o leer dentro de la memoria, las entradas y salidas de datos definen los datos a escribir y leer respectivamente, la entrada  $WE'$  controla el tipo de operación que la memoria debe hacer y la entrada  $OE'$  corresponde a la señal de habilitación de la memoria, la cual habilita o deshabilita la memoria para responder a las demás entradas.

En los computadores modernos las memorias actúan directamente con la *CPU* a través de canales de comunicación llamados buses. En la figura 181 se observa un esquema general, el cuál muestra cómo interactúa la *CPU* de un microcomputador con las memorias a través de estos canales.



**Figura 181.** Esquema Simplificado de un Microcomputador.

Las operaciones básicas de una memoria consisten en leer y almacenar información mediante el uso del bus de datos y direcciones. Estas operaciones ocurren en un orden lógico, el cual se indica a continuación:

- Apuntar a la dirección de memoria que se desea leer o escribir mediante el uso del bus de direcciones

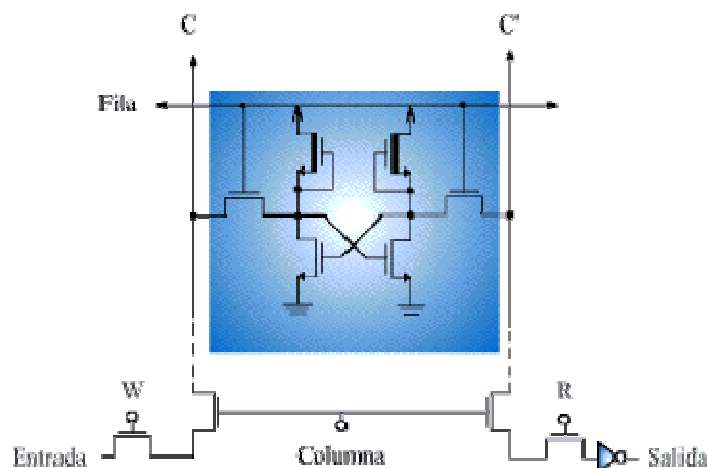
- Selección del tipo de operación: Lectura o escritura.
- Cargar los datos a almacenar (en el caso de una operación de escritura)
- Retener los datos de la memoria (en el caso de una operación de lectura)
- Habilitar o deshabilitar la memoria para una nueva operación.

### Memorias de Acceso Aleatorio

Las memorias de Acceso Aleatorio son conocidas como memorias *RAM* de la sigla en inglés *Random Access Memory*. Se caracterizan por ser memorias de lectura/escritura y contienen un conjunto de variables de dirección que permiten seleccionar cualquier dirección de memoria de forma directa e independiente de la posición en la que se encuentre.

Estas memorias son volátiles, es decir, que se pierde la información cuando no hay energía y se clasifican en dos categorías básicas: la *RAM estática* y la *RAM dinámica*, las cuales se describen en las siguientes dos secciones.

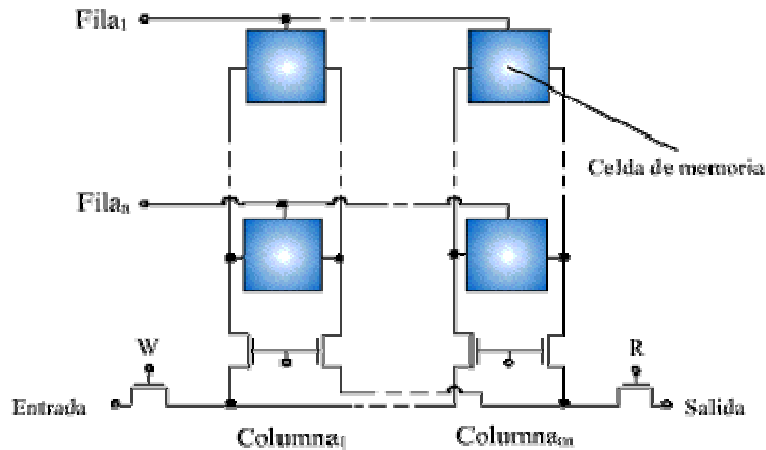
**Memoria RAM estática:** Este tipo de memoria conocida como *SRAM* (*Static Random Access Memory*) se compone de celdas conformadas por *FLIP-FLOPs* contruidos generalmente con transistores *MOSFET*, aunque también existen algunas memorias pequeñas contruidas con transistores bipolares. En la figura 182 se observa la estructura típica de una celda de memoria de una *SRAM*.



**Figura 182.** Estructura de una celda de memoria SRAM

Como se observa en la figura la celda se activa mediante un nivel activo a la entrada superior y los datos se cargan o se leen a través de las líneas laterales.

Las celdas de memoria se agrupan en filas y columnas para conformar el arreglo básico de la memoria. En la figura 183 se muestra la disposición de las celdas de memoria en una *SRAM*, donde se observa que cada una de las filas se habilita de forma simultánea para recibir o cargar los datos del *bus* de entrada/salida.



**Figura 183.** Arreglo básico de una *SRAM*

Existen otras modalidades de funcionamiento de las memorias estáticas, entre ellas se pueden nombrar las siguientes:

**SRAM Sincrónica:** Al igual que en los sistemas síncronos, este tipo de memoria tiene una entrada de reloj, la cual le permite operar en sincronía con otros dispositivos. Esta característica no aporta mejores beneficios, sin embargo simplifica enormemente el diseño de sistemas de alta prestaciones, ya que una única señal controla todos los dispositivos involucrados. La ventaja de estas memorias viene proporcionada por lo que se podría llamar su funcionamiento automático, guiado por la señal de reloj, por lo que no es necesario ocuparse de generar las señales de control.

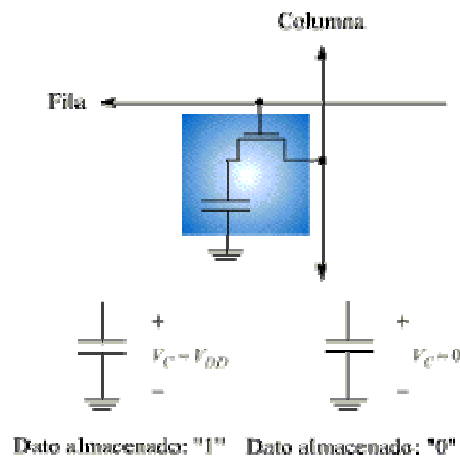
**SRAM de Ráfaga:** Las memorias de ráfagas (*burst*) son síncronas y se caracterizan por incluir un contador que permite que la memoria genere internamente la dirección a la que debe acceder, consiguiendo de esta forma accesos hasta cuatro posiciones de memoria con una sola dirección de referencia. Esto permite acceder de forma más rápida a la información en memoria.

Vemos como funciona este tipo de memoria. La *CPU* genera una dirección de memoria, la cual se propaga a través del *bus* de direcciones hasta la memoria, decodificándose y accediendo a la posición correspondiente. Si se ha configurado la memoria en modo ráfaga, una vez obtenido el primer dato, incrementa la dirección y vuelve a acceder. De esta forma se evita el tiempo de propagación de las señales por el *bus* y el tiempo de decodificación de la dirección. El número de palabras leídas o escritas en una ráfaga, viene limitado por el tamaño del contador interno de la memoria.

**SRAM Pipeline:** Con los dos tipos de memorias anteriores se consigue el acceso a posiciones consecutivas de forma rápida. Para mantener esta velocidad cuando se cambia de secuencia, las memorias *pipeline* incluyen un *buffer* para almacenar la dirección y los datos actuales proporcionados por la memoria. De esta forma, se puede enviar la nueva dirección antes de terminar la lectura, consiguiendo así que la *CPU* no espere la finalización del acceso a una posición de memoria para generar la nueva dirección.

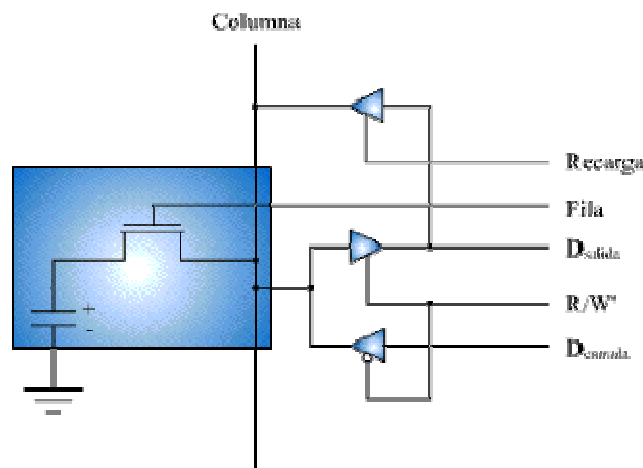
**Memoria RAM dinámica:** Este tipo de memoria conocida como *DRAM* (*Dinamic Random Access Memory*), a diferencia de la memoria estática se compone de celdas de memoria construidas con condensadores. Las celdas de memoria son de

fabricación más sencillas en comparación a las celdas a base de transistores, lo cual permite construir memorias de gran capacidad. La figura 184 se observa la composición interna de una de estas celdas.



**Figura 184.** Celda de memoria de una DRAM

La operación de la celda es similar a la de un interruptor, cuando el estado en la fila se encuentra en alto, el transistor entra en saturación y el dato presente en el bus interno de la memoria (columna) se almacena en el condensador, durante una operación de escritura y se extrae en una operación de lectura. El inconveniente que tiene este tipo de memorias consiste en que hay que recargar la información almacenada en las celdas, por lo cual estas celdas requieren de circuitería adicional para cumplir esta función. En la figura 185 se observa la celda completa con sus aditamentos donde se puede identificar la forma en que se desarrollan las operaciones de escritura, lectura y recarga.



**Figura 185.** Sistema lectura, escritura y recarga de una celda DRAM

Como se ha notado, existen diferencias entre la memoria Estática y Dinámica. En La tabla 76 se indican las ventajas y desventajas de los dos sistemas de memoria, lo cual permite seleccionar el tipo de memoria dependiendo de la aplicación.



Memoria	Ventajas	Desventajas
SRAM	<ul style="list-style-type: none"> <li>La velocidad de acceso es alta.</li> <li>Para retener los datos solo necesita estar energizada.</li> <li>Son mas fáciles de diseñar.</li> </ul>	<ul style="list-style-type: none"> <li>Menor capacidad, debido a que cada celda de almacenamiento requiere mas transistores.</li> <li>Mayor costo por <i>bit</i>.</li> <li>Mayor consumo de Potencia.</li> </ul>
DRAM	<ul style="list-style-type: none"> <li>Mayor densidad y capacidad.</li> <li>Menor costo por bit.</li> <li>Menor consumo de potencia.</li> </ul>	<ul style="list-style-type: none"> <li>La velocidad de acceso es bajar.</li> <li>Necesita recargar de la información. almacenada para retenerla.</li> <li>Diseño complejo.</li> </ul>

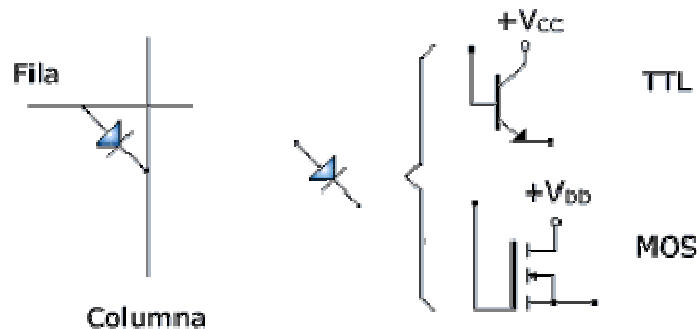
**Tabla 76.** Ventajas y desventajas de los dos sistemas de memoria

**Memorias de Solo Lectura:** Las memorias de solo lectura son conocidas como memorias *ROM* de la sigla en inglés *Read Only Memory*. Se caracterizan por ser memorias de lectura y contienen celdas de memoria no volátiles, es decir que la información almacenada se conserva sin necesidad de energía. Este tipo de memoria se emplea para almacenar información de forma permanente o información que no cambie con mucha frecuencia.

Actualmente se dispone de varios tipos de memorias *ROM*, a continuación se explicará cada una de ellas con sus características básicas.

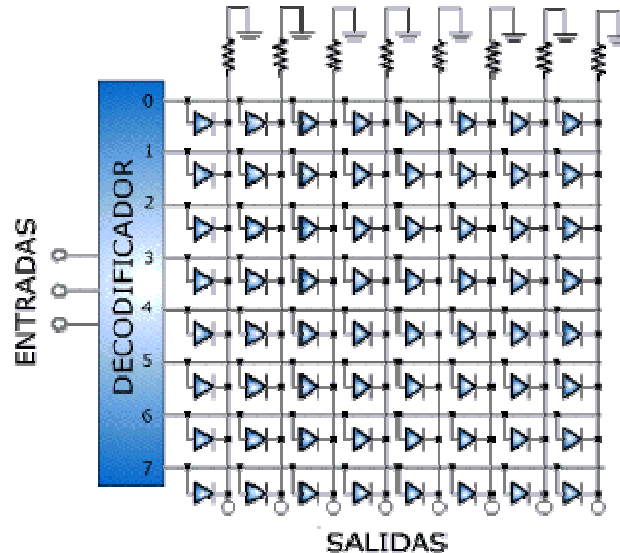
**Memoria ROM de Máscara:** Esta memoria se conoce simplemente como *ROM* y se caracteriza porque la información contenida en su interior se almacena durante su construcción y no se puede alterar. Son memorias ideales para almacenar microprogramas, sistemas operativos, tablas de conversión y caracteres.

Generalmente estas memorias utilizan transistores *MOS* para representar los dos estados lógicos (*1* ó *0*). La programación se desarrolla mediante el diseño de un negativo fotográfico llamado máscara donde se especifican las conexiones internas de la memoria. En la figura 186 se muestra la celda de memoria de una *ROM* de este tipo, en tecnologías *TTL* y *MOS*.



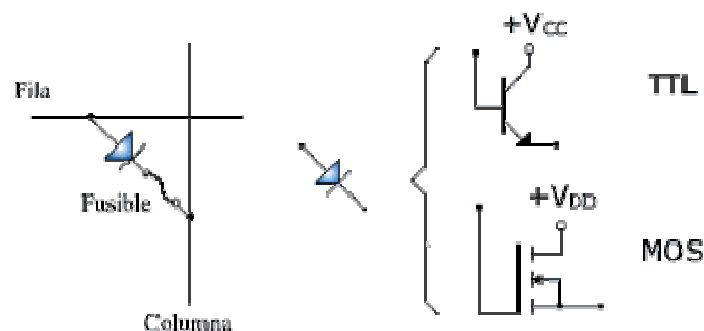
**Figura 186.** Celdas de memoria para una ROM

Las celdas de memoria se organizan en grupos para formar registros del mismo tamaño y estos se ubican físicamente formando un arreglo, como el indicado en la figura 187.

**Figura 187.** Organización interna de una Memoria ROM

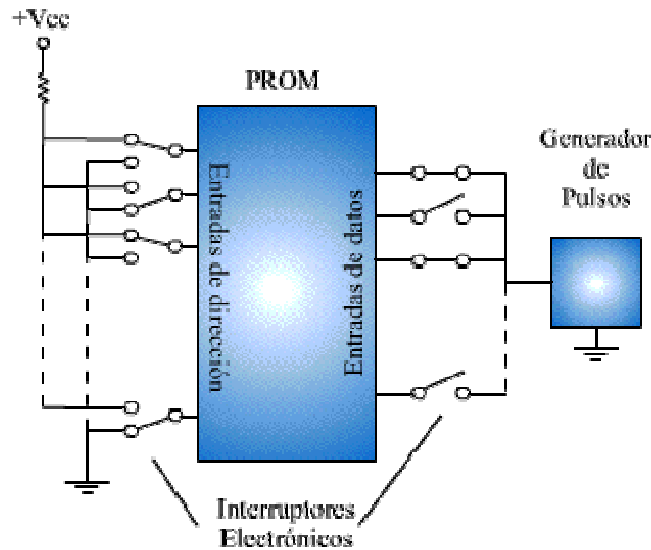
**Memoria PROM:** Esta memoria es conocida como ROM programable de la sigla en inglés *Programmable Read Only Memory*. Este tipo de memoria a diferencia de la ROM no se programa durante el proceso de fabricación, en vez de ello la programación la efectúa el usuario y se puede realizar una sola vez, después de la cual no se puede borrar o volver a almacenar otra información.

El proceso de programación es destructivo, es decir, que una vez grabada, es como si fuese una ROM normal. Para almacenar la información se emplean dos técnicas: por destrucción de fusible o por destrucción de unión. Comúnmente la información se programa o quema en las diferentes celdas de memoria aplicando la dirección en el bus de direcciones, los datos en los buffers de entrada de datos y un pulso de 10 a 30V, en una terminal dedicada para fundir los fusibles correspondientes. Cuando se aplica este pulso a un fusible de la celda, se almacena un 0 lógico, de lo contrario se almacena un 1 lógico (estado por defecto), quedando de esta forma la información almacenada de forma permanente. En la figura 188 se observa la disposición interna de una celda de memoria y los fusibles correspondientes.

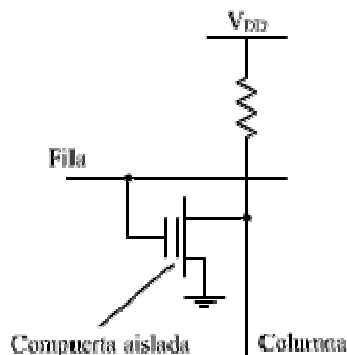


**Figura 188.** Celda de Memoria de una *PROM*

El proceso de programación de una *PROM* generalmente se realiza con un equipo especial llamado quemador. Este equipo emplea un mecanismo de interruptores electrónicos controlados por software que permiten cargar las direcciones, los datos y genera los pulsos para fundir los fusibles del arreglo interno de la memoria. En la figura 189 se indica de forma esquemática la función del programador.

**Figura 189.** Programación de un *PROM*

**Memoria EPROM:** Este tipo de memoria es similar a la *PROM* con la diferencia que la información se puede borrar y volver a grabar varias veces. Su nombre proviene de la sigla en inglés *Erasable Read Only Memory*. La programación se efectúa aplicando en un *pin* especial de la memoria una tensión entre 10 y 25 Voltios durante aproximadamente 50 ms, según el dispositivo, al mismo tiempo se direcciona la posición de memoria y se pone la información a las entradas de datos. Este proceso puede tardar varios minutos dependiendo de la capacidad de memoria. La memoria *EPROM*, tal como las memorias vistas anteriormente se compone de un arreglo de transistores *MOSFET* de Canal *N* de compuerta aislada. En la figura 190 se observa el transistor funcionando como celda de memoria en una *EPROM*.

**Figura 190.** Celda de memoria de una *EPROM*

Cada transistor tiene una compuerta flotante de  $\text{SiO}_2$  (sin conexión eléctrica) que en estado normal se encuentra apagado y almacena un 1 lógico. Durante la

programación, al aplicar una tensión (10 a 25V) la región de la compuerta queda cargada eléctricamente, haciendo que el transistor se encienda, almacenando de esta forma un 0 lógico. Este dato queda almacenado de forma permanente, sin necesidad de mantener la tensión en la compuerta ya que la carga eléctrica en la compuerta puede permanecer por un período aproximado de 10 años.

Por otra parte el borrado de la memoria se realiza mediante la exposición del dispositivo a rayos ultravioleta durante un tiempo aproximado de 10 a 30 minutos. Este tiempo depende del tipo de fabricante y para realizar el borrado, el circuito integrado dispone de una ventana de cuarzo transparente, la cual permite a los rayos ultravioleta llegar hasta el material fotoconductor presente en las compuertas aisladas y de esta forma lograr que la carga se disipe a través de este material apagando el transistor, en cuyo caso todas las celdas de memoria quedan en 1 lógico. Generalmente esta ventana de cuarzo se ubica sobre la superficie del encapsulado y se cubre con un adhesivo para evitar la entrada de luz ambiente que pueda borrar la información, debido a su componente UV. En la figura 10.3.6 se observa la fotografía de una memoria de este tipo.



**Figura 191.** Apariencia Física de una EPROM

**Memoria EEPROM:** La memoria *EEPROM* es programable y borrrable eléctricamente y su nombre proviene de la sigla en inglés *Electrical Erasable Programmable Read Only Memory*. Actualmente estas memorias se construyen con transistores de tecnología MOS (*Metal Oxide Silice*) y MNOS (*Metal Nitride-Oxide Silicon*).

Las celdas de memoria en las *EEPROM* son similares a las celdas *EPROM* y la diferencia básica se encuentra en la capa aislante alrededor de cada compuesta flotante, la cual es más delgada y no es fotosensible.

La programación de estas memorias es similar a la programación de la *EPROM*, la cual se realiza por aplicación de una tensión de 21 Voltios a la compuerta aislada MOSFET de cada transistor, dejando de esta forma una carga eléctrica, que es suficiente para encender los transistores y almacenar la información. Por otro lado, el borrado de la memoria se efectúa aplicando tensiones negativas sobre las compuertas para liberar la carga eléctrica almacenada en ellas.

Esta memoria tiene algunas ventajas con respecto a la Memoria *EPROM*, de las cuales se pueden enumerar las siguientes:

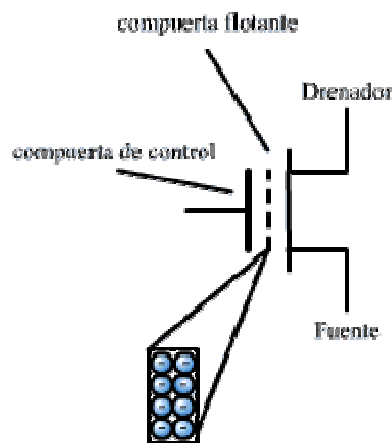
- Las palabras almacenadas en memoria se pueden borrar de forma individual.
- Para borrar la información no se requiere luz ultravioleta.
- Las memorias *EEPROM* no requieren programador.

- Para reescribir no se necesita se necesita hacer un borrado previo.
- Se pueden reescribir aproximadamente unas 1000 veces sin que se observen problemas para almacenar la información.

El tiempo de almacenamiento de la información es similar al de las *EPROM*, es decir aproximadamente 10 años.

**Memoria FLASH:** La memoria *FLASH* es similar a la *EEPROM*, es decir que se puede programar y borrar eléctricamente. Sin embargo esta reúne algunas de las propiedades de las memorias anteriormente vistas, y se caracteriza por tener alta capacidad para almacenar información y es de fabricación sencilla, lo que permite fabricar modelos de capacidad equivalente a las *EPROM* a menor costo que las *EEPROM*.

Las celdas de memoria se encuentran constituidas por un transistor MOS de puerta apilada, el cual se forma con una puerta de control y una puerta aislada, tal como se indica en la figura 192. La compuerta aislada almacena carga eléctrica cuando se aplica una tensión lo suficientemente alta en la puerta de control. De la misma manera que la memoria *EPROM*, cuando hay carga eléctrica en la compuerta aislada, se almacena un 0, de lo contrario se almacena un 1.



**Figura 192.** Celda de memoria de una FLASH

Las operaciones básicas de una memoria *Flash* son la programación, la lectura y borrado.

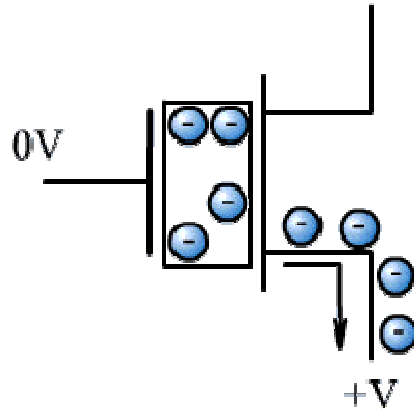
Como ya se mencionó, la programación se efectúa con la aplicación de una tensión (generalmente de 12V o 12.75 V) a cada una de las compuertas de control, correspondiente a las celdas en las que se desean almacenar 0's. Para almacenar 1's no es necesario aplicar tensión a las compuertas debido a que el estado por defecto de las celdas de memoria es 1.

La lectura se efectúa aplicando una tensión positiva a la compuerta de control de la celda de memoria, en cuyo caso el estado lógico almacenado se deduce con base en el cambio de estado del transistor:

- Si hay un 1 almacenado, la tensión aplicada será lo suficiente para encender el transistor y hacer circular corriente del drenador hacia la fuente.
- Si hay un 0 almacenado, la tensión aplicada no encenderá el transistor debido a que la carga eléctrica almacenada en la compuerta aislada.

Para determinar si el dato almacenado en la celda es un *1* ó un *0*, se detecta la corriente circulando por el transistor en el momento que se aplica la tensión en la compuerta de control.

El borrado consiste en la liberación de las cargas eléctricas almacenadas en las compuertas aisladas de los transistores. Este proceso consiste en la aplicación de una tensión lo suficientemente negativa que desplaza las cargas como se indica en la figura 193.



**Figura 193.** Proceso de descarga de una celda de memoria *FLASH*

**7.7. Aplicaciones de las Memorias:** En la actualidad muchos de los sistemas electrónicos necesitan dispositivos para almacenar y/o leer información. Como ejemplo de este tipo de sistemas podemos mencionar los teléfonos electrónicos, televisores, equipos de sonido y los computadores entre otros.

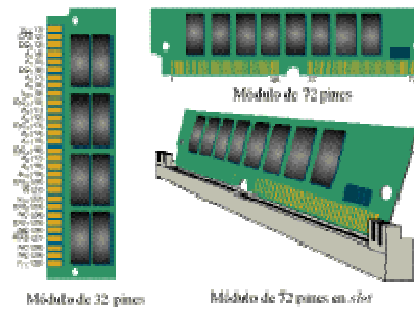
Esta lección se centrará en mencionar algunas aplicaciones particulares de las memorias que pueden ser de interés para desarrollar proyectos.

**Memoria RAM:** La memoria *RAM* es uno de los componentes más importantes en un computador. Cuando se requiere emplear un archivo de datos o programa, los datos o instrucciones son leídos desde el disco duro o disquete y colocados en una memoria *RAM*, para que sean leídos por el microprocesador, permitiéndole manipularlos, ingresar nuevos datos, modificar los existentes, hacer cálculos, búsquedas, resúmenes, etc.

El uso más difundido de estos dispositivos indiscutiblemente se encuentra en los computadores:

- Se utilizan en sistemas microprocesados, y en los microcontroladores, en sistemas pequeños es de lectura/escritura.
- En los computadores se utiliza como memoria de Cache y memoria de vídeo.

Las memorias para computadores generalmente no se consiguen en *chips*, sino en módulos de memoria empaquetados en dos formatos básicos: *SIMM* y *DIMM* que contienen 8, 16, 32, 64 o 128 *MB* cada uno. Estos módulos se introducen en ranuras o *slots* en la tarjeta madre de los computadores y en la figura 194 se muestra su presentación de 32 y 72 pines.



**Figura 194.** Módulos de Memoria RAM

### Memoria ROM:

#### Programas y Datos

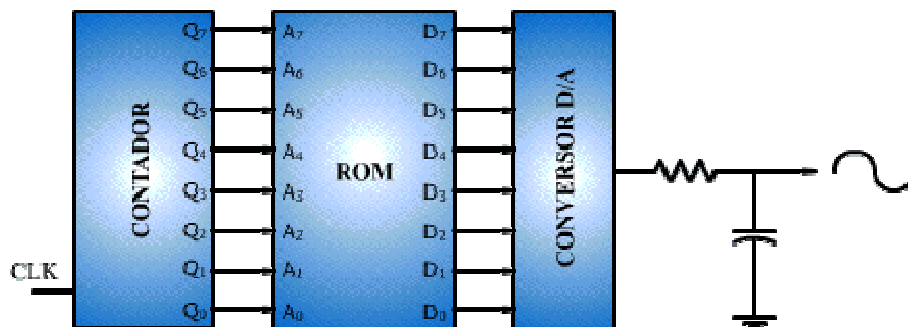
La aplicación más difundida en este tipo de memoria ha sido el almacenamiento de códigos de programas para el momento del arranque de dispositivos que utilizan microprocesadores, como es el caso de los computadores.

Los Computadores vienen con una memoria ROM, donde se encuentran alojados los programas del BIOS (*Basic Input Output System*), el cual contiene las instrucciones y datos necesarios para activar y hacer funcionar el computador y sus periféricos. Debido a que en esta memoria la información está disponible en cualquier momento, los programas en una ROM son a menudo los cimientos sobre los que se construye el resto de los programas (incluyendo el DOS).

La ventaja de tener los programas fundamentales del computador almacenados en una ROM, es que están allí disponibles y no hay necesidad de cargarlos en la memoria desde el disco de la misma forma que se carga el DOS. Comúnmente estos programas son llamados *Firmware*, lo que indica que se encuentran firmemente almacenados en el Hardware y que no cambian.

#### Funciones matemáticas y Generadores de Señales

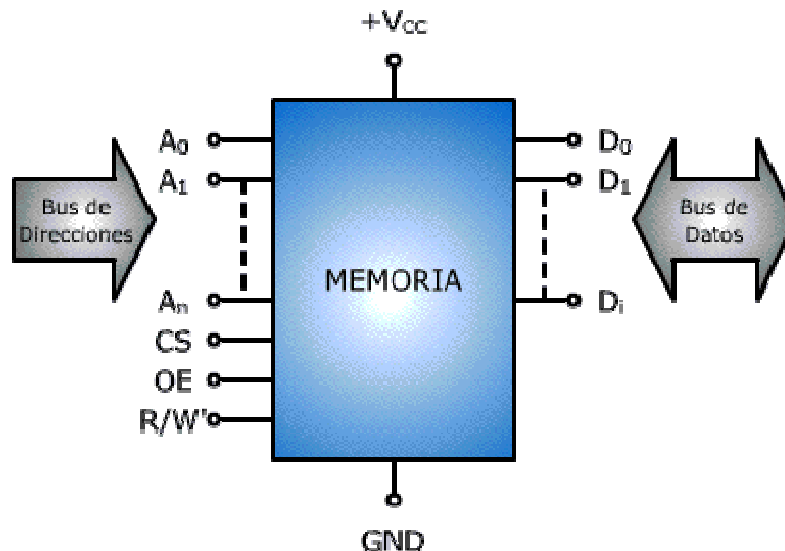
Otra aplicación de estas memorias consiste en el almacenamiento de tablas de datos que permiten generar funciones matemáticas. Por ejemplo existen memorias que almacenan funciones trigonométricas y hallan el resultado con base en el valor binario introducido en el bus de direcciones. En la figura 195, se observa como se puede implementar un generador de una señal seno, a partir de la información almacenada en una memoria ROM.



**Figura 195.** Generador de señales con una memoria ROM

### Ejemplos de Memorias Comerciales

Las memorias son circuitos integrados cuyos pines se hayan en ambos lados de la cápsula, formando dos líneas o hileras de pines (*DIP*) y generalmente se fabrican con capacidades de orden de Kilobytes o Megabytes múltiplos de 8, por ejemplo 8k, 16k, 32k, 64k, 128k, o 8M, 16M, 32M, etc.



**Figura 196.** Distribución de pines de un chip de memoria

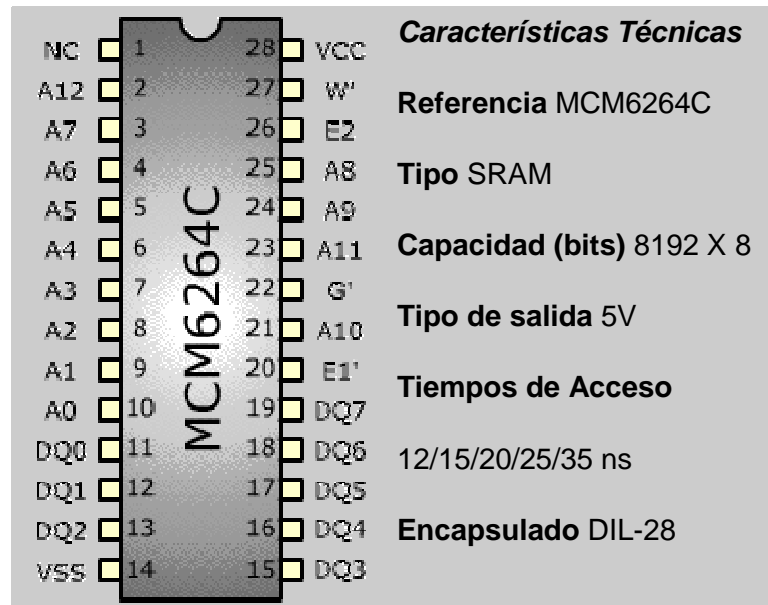
En la figura 196 se observa un esquema descriptivo de los pines que generalmente se encuentran en una memoria. A continuación se da una explicación de cada uno de estos pines:

- **$A_0...A_n$  (Bus de direcciones):** Estos pines son las entradas para seleccionar la posición de memoria a escribir o leer y su cantidad define la capacidad de palabras que puede almacenar, dada por la expresión  $2^n$ , donde  $n$  es el número de pines.
- **$D_0...D_i$  (Bus de Datos):** Corresponde a los pines de entrada y salida de datos. En el mercado se consiguen generalmente buses de 1, 4, 8 y 16 *bits* y lo más usual es encontrar *chips* tengan 8 entradas de datos.
- **CS (Chip Select):** Este *pin* se utiliza para seleccionar el chip de memoria que se desea acceder. Esto en el caso del usar dos o más memorias similares.
- **OE (Output Enable):** Utilizado para habilitar la salida de datos. Cuando se encuentra en estado activo las salidas tiene alta impedancia o actúan como entradas.
- **R/W' (Read/Write):** Entrada utilizada en las memorias RAM para seleccionar la operación de lectura o escritura
- **$V_{cc}$  y GND (Alimentación):** Corresponden a los pines de alimentación del circuito integrado. Algunas tienen disponible tres pines para este propósito, pero por lo general son dos y el valor de la tensión de alimentación depende de la tecnología de fabricación del circuito.

En las siguientes secciones se indicaran algunos ejemplos de circuitos integrados de uso general disponibles en el mercado, dando un ejemplo de cada uno de los tipos de memorias vistas.

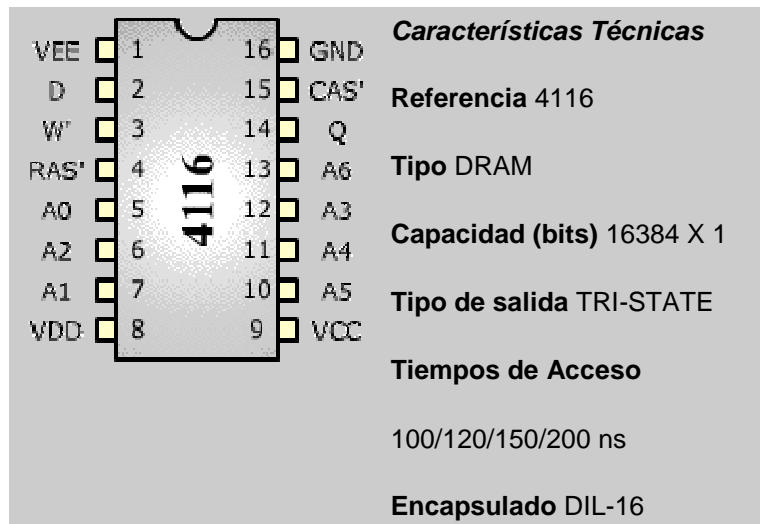


**MEMORIA SRAM - MCM6264C:** Esta memoria fabricada por *Motorola* y desarrollada con tecnología *CMOS* tiene una capacidad de 8K x 8. Los tiempos de lectura y escritura del integrado son de aproximadamente 12 ns y tiene un consumo de potencia aproximado de 100 mW. En la Figura 197 se observa la disposición de los pines del circuito integrado de esta memoria y sus las características técnicas básicas.



**Figura 197.** SRAM MCM6264C

**MEMORIA DRAM – 4116:** El *CI 4116* es una memoria *DRAM* de 16K x 1. La estructura interna de este integrado se encuentra constituida por un arreglo de 128 filas y 128 columnas donde cada uno de los *bits* se ubica con una dirección de 14 *bits*. En la figura 198 se muestra la disposición de los pines del circuito integrado. Observe que la entrada de direcciones es de 7 *bits* (*A0...A6*). La razón de poseer 7 pines y no 14, se debe a que estos tienen función doble, por ejemplo la entrada *A0* se utiliza para establecer los valores de los *bits A0/A7* de la dirección de memoria que se quiere acceder.



**Figura 198.** DRAM 4116

Para ingresar una dirección de memoria en este integrado se utilizan las señales de entrada *RAS'* y *CAS'*, las cuales deben estar inicialmente en "1" para recibir los 7 bits menos significativos de la dirección (*A6...A0*). Después de ello la entrada *RAS'* debe cambiar a "0" con lo cual los 7 bits se cargan en el registro de direcciones de memoria y el dispositivo queda disponible para recibir los 7 bits mas significativos (*A7...A14*) de la dirección. Una vez se aplican estos *bits*, la entrada *CAS'* debe cambiar a "0", cargándolos de esta forma en el registro de direcciones en su respectiva posición y permitiendo finalmente acceder a la posición de memoria para efectuar la operación de lectura o escritura.

**MEMORIAS PROM - 74S473:** Esta memoria tiene una capacidad de 512 palabras de 8 *bits* y la descripción de sus pines se muestra en la figura 199

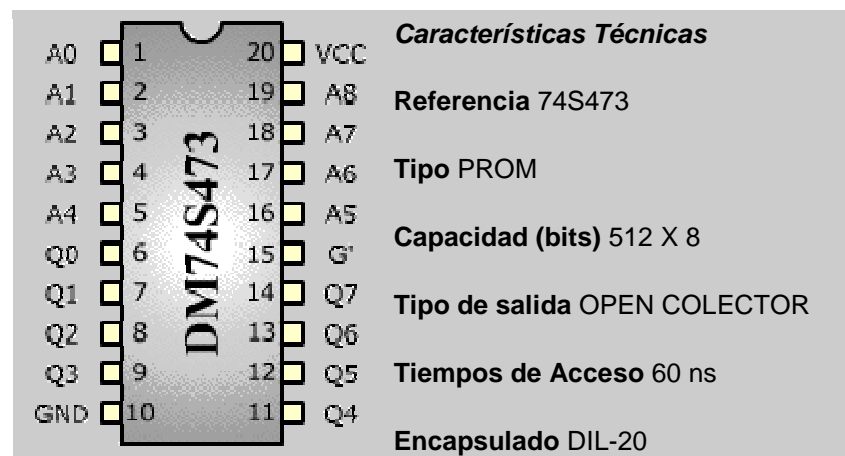


Figura 199. PROM 74S473

**MEMORIA EPROM - 27C16B:** Esta memoria de 24 pines tiene una capacidad de 2048 palabras de 8 *bits*, es decir 2KB. Las salidas de esta memoria son triestado, lo que permite escribir o leer los datos con el mismo bus de datos.

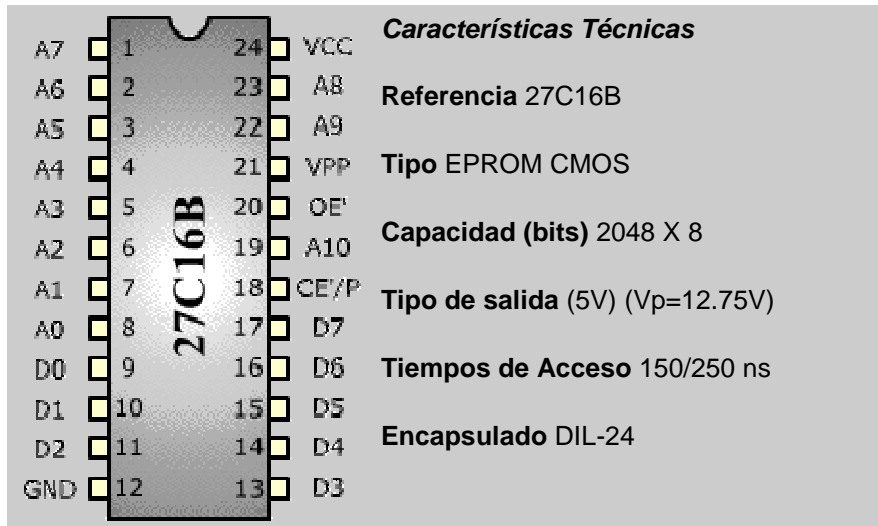


Figura 200. EPROM 27C16B

Esta memoria tiene dos pines no indicados inicialmente:

- VPP: Es utilizado durante la programación.
- CE'/P (Chip Enable'/Program): Utilizado para seleccionar el chip (en caso de emplearse en forma conjunta con otros) y para programar la posición de memoria seleccionada en el bus de direcciones.

Durante la programación de la memoria, la entrada  $OE'$  se debe encontrar en 1. En la entrada debe estar presente una tensión de 5V, así como en los datos y la dirección de memoria. Después de ello, se aplica pulso de tensión durante 30 ms aproximadamente, para almacenar los datos.

Como se vió anteriormente, el borrado de este tipo de memoria se efectúa mediante la exposición del integrado a luz ultravioleta. Una lámpara UV de 12mW, puede ser utilizada para efectuar este proceso, el cual tarda entre 20 y 25 minutos.

**MEMORIA EEPROM - 28C64A:** Esta memoria tiene una capacidad de 8K X 8 y tiene características diferentes a las demás. La información almacenada puede perdurar aproximadamente 100 años y puede soportar hasta 100.000 ciclos de grabado y borrado.

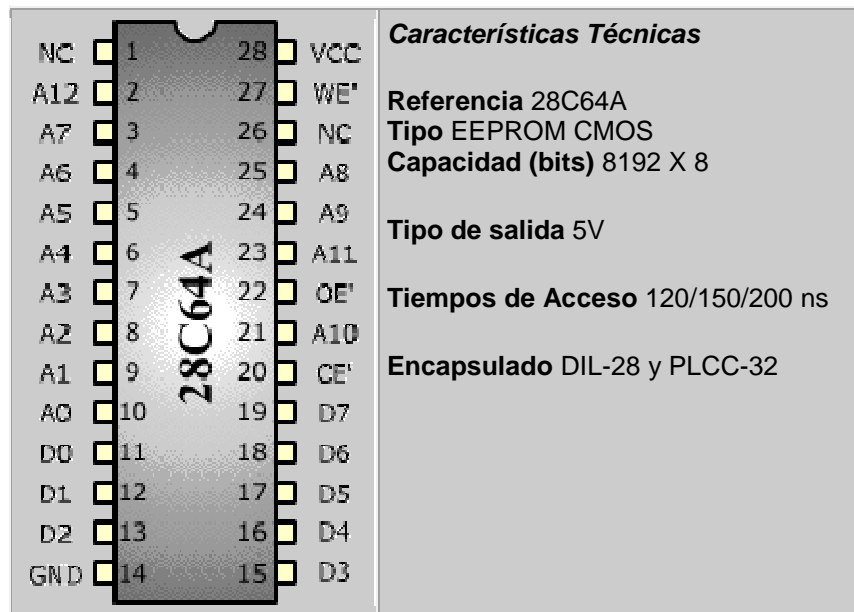


Figura 201. EEPROM 28C64A

En la figura 201 se indica la disposición de los pines de esta memoria la cual se encuentra disponible en dos tipos de encapsulados (DIL y PLCC).

**MEMORIA FLASH - 27F256:** La capacidad de esta memoria es de 32K X 8 y como memoria *Flash* tiene la característica particular de ser borrada en un tiempo muy corto (1 seg.). El tiempo de programación por byte es de 100 ms y el tiempo de retención de la información es de aproximadamente 10 años.

En la figura 202 se indica la disposición de los pines de esta memoria con sus características técnicas básicas.

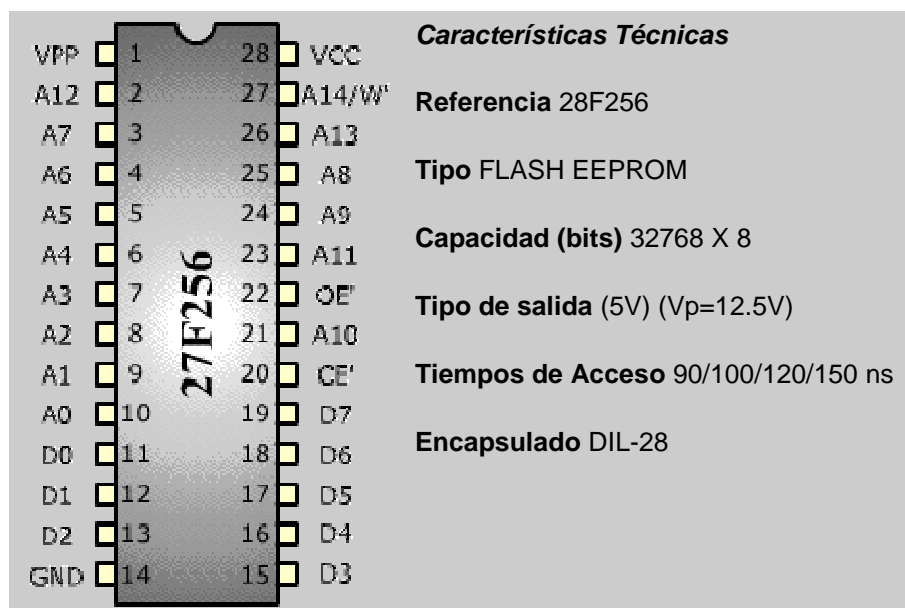


Figura 202. Memoria Flash 27F256

## 7.8. Lógica programable temprana

En 1970, Texas Instruments desarrolló un circuito integrado (CI) de máscara programable basado en la memoria asociativa de sólo lectura (*ROAM*) de IBM. Este dispositivo, el TMS2000, era programado alterando la capa metálica durante la producción del CI. El TMS2000 tenía hasta 17 entradas y 18 salidas con 8 biestables JK como memoria. Texas Instruments acuñó el término *Programmable logic array* para este dispositivo.

En 1973 National Semiconductor introdujo un dispositivo PLA de máscara programable (DM7575) con 14 entradas y 8 salidas sin registros de memoria. Este era más popular que el de Texas Instruments, pero el coste de hacer la máscara metálica limitaba su uso. El dispositivo es significativo por ser la base de la *FPGA (Field Programmable Logic Array)* producido por Signetics en 1975, el 82S100.

En 1971, General Electric desarrollaba un PLD basado en la nueva tecnología PROM. Este dispositivo experimental mejoró el ROAM de IBM permitiéndole realizar lógica multinivel. Intel acababa de introducir la PROM de puerta flotante borrable por Ultravioleta por lo que el desarrollador en General Electric incorporó esa tecnología. El dispositivo de General Electric era el primer PLD jamás desarrollado, antecesora del EPLD de Altera en una década. General Electric obtuvo varias patentes tempranas en PLDs.

En 1974, General Electric firmó un acuerdo con Monolithic Memories para desarrollar un PLD de máscara programable incorporando las innovaciones de General. El dispositivo se bautizó como *Programmable Associative Logic Array* (PALA, matriz lógica asociativa programable). El MMI 5760 fue terminado en 1976 y podía implementar circuitos multinivel o secuenciales de más de 100 puertas. El dispositivo estaba soportado por el entorno de desarrollo de General, donde las ecuaciones Booleanas podían ser convertidas a patrones de máscara para configurar el dispositivo. El integrado nunca se comercializó.

**CPLDs:** Las PALs y GALs están disponibles sólo en tamaños pequeños, equivalentes a unos pocos cientos de puertas lógicas. Para circuitos lógicos mayores, se pueden utilizar PLDs complejos o CPLDs. Estos contienen el equivalente a varias PAL enlazadas por interconexiones programables, todo ello en el mismo circuito integrado. Las CPLDs pueden reemplazar miles, o incluso cientos de miles de puertas lógicas. Algunas CPLDs se programan utilizando un programador PAL, pero este método no es manejable para dispositivos con cientos de pines. Un segundo método de programación es soldar el dispositivo en su circuito impreso. Las CPLDs contienen un

circuito que decodifica la entrada de datos y configura la CPLD para realizar su función lógica específica. Cada fabricante tiene un nombre propietario para este sistema de programación. Por ejemplo, Lattice Semiconductor la llama in-system programming (programación dentro del sistema). Sin embargo, estos sistemas propietarios están dejando paso al estándar del Joint Test Action Group (JTAG).

**FPGAs** Field programmable gate array: Mientras el desarrollo de las PALs se enfocaba hacia las GALs y CPLDs, apareció una corriente de desarrollo distinta. Esta corriente de desarrollo desembocó en un dispositivo basado en la tecnología de matriz de puertas y se le denominó *field-programmable gate array* (FPGA). Algunos ejemplos de las primeras FPGAs son la matriz 82s100 y el secuenciador 82S105 de Signetics, presentados a finales de los 70. El 82S100 era una matriz de términos AND, y también tenía funciones de Flip-Flops..

Las FPGAs utilizan una rejilla de puertas lógicas, similar a la de una matriz de puertas ordinarias, pero la programación en este caso la realiza el cliente, no el fabricante. El término field-programmable (literamente programable en el campo) se refiere a que la matriz se define fuera de la fábrica, o "en el campo".

Una FPGA (del inglés Field Programmable Gate Array) es un dispositivo semiconductor que contiene bloques de lógica cuya interconexión y funcionalidad se puede programar. La lógica programable puede reproducir desde funciones tan sencillas como las llevadas a cabo por una puerta lógica o un sistema combinacional hasta complejos sistemas en un chip (we:en:System-on-a-chip).

Las FPGAs se utilizan en aplicaciones similares a los ASICs sin embargo son más lentas, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGAs tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), sus costes de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.

Ciertos fabricantes cuentan con FPGAs que sólo se pueden programar una vez, por lo que sus ventajas e inconvenientes se encuentran a medio camino entre los ASICs y las FPGAs reprogramables.

Históricamente las FPGAs surgen como una evolución de los conceptos desarrollados en las PLAs y los CPLDs

---

**Historia: FPGAs vs CPLDs:** Las FPGAs son inventadas en el año 1984 por Ross Freeman, co-fundador de Xilinx, y surgen como una evolución de los CPLDs.

Tanto los CPLDs como las FPGAs contienen un gran número de elementos lógicos programables. Si medimos la densidad de los elementos lógicos programables en puertas lógicas equivalentes (numero de puertas NAND equivalentes que podríamos programar en un dispositivo) podríamos decir que en un CPLD hallaríamos del orden de decenas de miles de puertas lógicas equivalentes y en una FPGA del orden de cientos de miles hasta millones de ellas.

Aparte de las diferencias en densidad entre ambos tipos de dispositivos, la diferencia fundamental entre las FPGAs y los CPLDs es su arquitectura. La arquitectura de los CPLDs es más rígida y consiste en una o más sumas de productos programables cuyos resultados van a parar a un número reducido de biestables síncronos (también denominados flip-flops). La arquitectura de las FPGAs, por otro lado, se basa en un gran número de pequeños bloques utilizados para reproducir sencillas operaciones lógicas, que cuentan a su vez con biestables síncronos. La enorme libertad disponible en la interconexion de dichos bloques confiere a las FPGAs una gran flexibilidad.

Otra diferencia importante entre FPGAs y CPLDs es que en la mayoría de las FPGAs se pueden encontrar funciones de alto nivel (como sumador y multiplicador) embebidas en la propia matriz de interconexiones, así como bloques de memoria.

**Características FPGA:** Una jerarquía de interconexiones programables permite a los bloques lógicos de un FPGA ser interconectados según la necesidad del diseñador del sistema, algo parecido a un breadboard programable. Estos bloques lógicos e interconexiones pueden ser programados después del proceso de manufactura por el usuario/diseñador, así que el FPGA puede desempeñar cualquier función lógica necesaria.

Una tendencia reciente ha sido combinar los bloques lógicos e interconexiones de los FPGA con microprocesadores y periféricos relacionados para formar un «Sistema programable en un chip». Ejemplo de tales tecnologías híbridas pueden ser encontradas en los dispositivos Virtex-II PRO y Virtex-4 de Xilinx, los cuales incluyen uno o más procesadores PowerPC embebidos junto con la lógica del FPGA. El FPSLIC de Atmel es otro dispositivo similar, el cual usa un procesador AVR en combinación con la arquitectura lógica programable de Atmel. Otra alternativa es hacer uso de núcleos de procesadores implementados haciendo uso de la lógica del FPGA. Esos núcleos incluyen los procesadores MicroBlaze y PicoBlaze de Xilinx, Nios y Nios II de Altera, y los procesadores de código abierto LatticeMicro32 y LatticeMicro8.

---

Muchos FPGA modernos soportan la reconfiguración parcial del sistema, permitiendo que una parte del diseño sea reprogramada, mientras las demás partes siguen funcionando. Este es el principio de la idea de la «computación reconfigurable», o los «sistemas reconfigurables».

**Programación FPGA:** La tarea del programador es definir la función lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos.

El diseñador cuenta con la ayuda de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación especiales son conocidos como HDL o Hardware Description Language (lenguajes de descripción de hardware). Los HDLs más utilizados son: VHDL, Verilog, ABEL. En un intento de reducir la complejidad y el tiempo de desarrollo en fases de prototipaje rápido, y para validar un diseño en HDL, existen varias propuestas y niveles de abstracción del diseño. Entre otras, National Instruments LabVIEW FPGA propone un acercamiento de programación gráfica de alto nivel.

**Aplicaciones FPGA:** Cualquier circuito de aplicación específica puede ser implementado en un FPGA, siempre y cuando esta disponga de los recursos necesarios. Las aplicaciones donde más comúnmente se utilizan los FPGA incluyen a los DSP (procesamiento digital de señales), radio definido por software, sistemas aeroespaciales y de defensa, prototipos de ASICs, sistemas de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, emulación de hardware de computadora, entre otras. Cabe notar que su uso en otras áreas es cada vez mayor, sobre todo en aquellas aplicaciones que requieren un alto grado de paralelismo.

Existe código fuente disponible (bajo licencia GNU GPL)<sup>1</sup> de sistemas como microprocesadores, microcontroladores, filtros, módulos de comunicaciones y memorias, entre otros. Estos códigos se llaman cores.

**Fabricantes FPGA:** A principios de 2007, el mercado de los FPGA se ha colocado en un estado donde hay dos productores de FPGA de propósito general que están a la cabeza del mismo, y un conjunto de otros competidores quienes se diferencian por ofrecer dispositivos de capacidades únicas.

---



Xilinx es uno de los dos grandes líderes en la fabricación de FPGA.

Altera es el otro gran líder.

Lattice Semiconductor lanzó al mercado dispositivos FPGA con tecnología de 90nm. En adición, Lattice es un proveedor líder en tecnología no volátil, FPGA basadas en tecnología Flash, con productos de 90nm y 130nm.

Actel tiene FPGAs basados en tecnología Flash reprogrammable. También ofrece FPGAs que incluyen mezcladores de señales basados en Flash.

QuickLogic tiene productos basados en antifusibles (programables una sola vez).

Atmel es uno de los fabricantes cuyos productos son reconfigurables (el Xilinx XC62xx fue uno de estos, pero no están siendo fabricados actualmente). Ellos se enfocaron en proveer microcontroladores AVR con FPGAs, todo en el mismo encapsulado.

Achronix Semiconductor tienen en desarrollo FPGAs muy veloces. Planean sacar al mercado a comienzos de 2007 FPGAs con velocidades cercanas a los 2GHz.

MathStar, Inc. ofrecen FPGA que ellos llaman FPOA (Arreglo de objetos de matriz programable).

---

**BIBLIOGRAFÍA**

RUEDA, Luis Tutorial de Electronica Digital. Argentina 2004. Archivo PDF Capstone course independent study program. Publicación: Ewing, N.J. Exploring Design & Engineering Project, 2000 Documento: Inglés (eng) User interface design for electronic appliances <http://www.netLibrary.com/urlapi.asp?action=summary&v=1&bookid=83298>

Baumann, Konrad,; Thomas, Bruce, Publicación: London ; New York : Taylor & Francis, 2001 Documento: Inglés (eng) : User interface design for electronic appliances

<http://site.ebrary.com/lib/albertaac/Doc?id=10071249>

<http://site.ebrary.com/lib/rdcollege/Doc?id=10071249> Baumann, Konrad,; Thomas, Bruce, Publicación: London : Taylor & Francis, 2002, 2001 Documento: Inglés (eng) : User interface design for electronic appliances

<http://www.contentreserve.com/TitleInfo.asp?ID={00AB336C-C8FA-4441-ACE0-41DE57F970F}&Format=50>

<http://www.contentreserve.com/TitleInfo.asp?ID={ABBFFCD1-6547-4567-BB88-29F12EA9A34D}&Format=50> Baumann, Konrad,; Thomas, Bruce, Publicación: London ; New York : Taylor & Francis, 2001 Documento: Inglés (eng) : The digital sublime myth, power, and cyberspace /

<http://www.netLibrary.com/urlapi.asp?action=summary&v=1&bookid=122533> Mosco, Vincent. Publicación: Cambridge, Mass. : MIT Press, 2004 Documento: Inglés (eng)

<http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/100301.htm>

<http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/100501.htm>

<http://www.monografias.com/trabajos12/mosscur/mosscur.shtml>

[http://www.zona-warez.com/tutoriales-ingenieria\\_electronica.html](http://www.zona-warez.com/tutoriales-ingenieria_electronica.html)

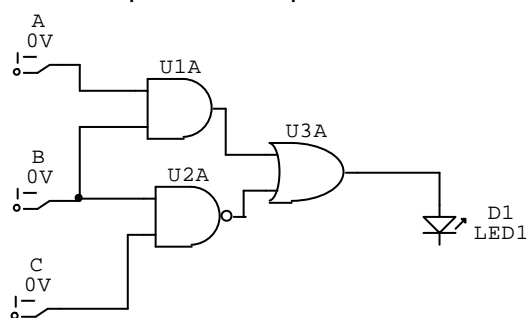
<http://electronred.iespana.es/electronred/Circuitosintegra.htm>

---

**ANEXO 1: Evaluación Diseño Electrónico Digital**

Selección múltiple Única Respuesta

1. El equivalente binario del numero decimal 27 es
  - a. 11011
  - b. 10111
  - c. 11001
  - d. 11101
2. El Equivalente binario del numero hexadecimal E7 es:
  - a. 11100111
  - b. 01111110
  - c. 01111111
  - d. 11110111
3. El Equivalente Octal del numero Binario 111000001 es:
  - a. 701
  - b. 103
  - c. 031
  - d. 013
4. La clasificación de los circuitos Integrados en: LSI, MSI, VLSI corresponde a:
  - a. Cantidad de pines que posee cada circuito integrado
  - b. Serie de Velocidad de cada circuito integrado dado por el fabricante
  - c. Serie de consumo de potencia da cada circuito integrado dado por el fabricante
  - d. Cantidad de transistores o compuertas que posee cada circuito integrado
5. Según el Teorema de D'Morgan  $(X+Y)'$  es equivalente a:
  - a.  $(XY)'$
  - b.  $X'Y'$
  - c.  $X'+Y'$
  - d.  $X'+Y$
6. La expresión algebraica implementada por circuito de la figura es:



- a.  $(AB)'+BC$
  - b.  $(A+B)'+(B+C)$
  - c.  $AB+(BC)'$
  - d.  $(A+B)(B+C)'$
7. La expresión algebraica simplificada por Minterminos empleando Mapas de Karnaugh de la figura es:

		BC			
		00	01	11	10
A	0	0	0	1	0
	1	1	1	1	1

- $A'(B'+C')$
- $A+BC$
- $A(B+C)$
- $A'+(BC)'$

8. La expresión algebraica simplificada por Maxterminos empleando Mapas de Karnaugh de la figura es:

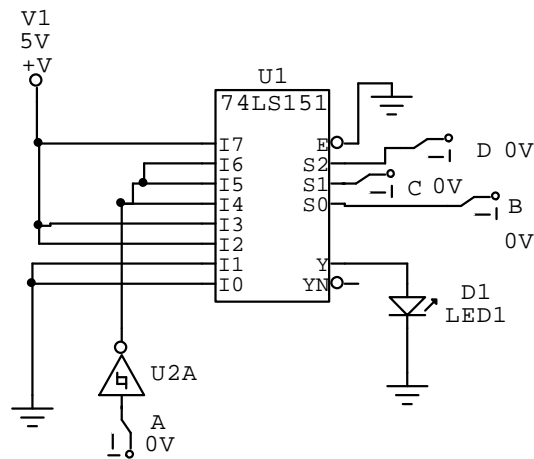
		BC			
		00	01	11	10
A	0	0	1	1	0
	1	0	1	1	0

- C
- C'
- B+C
- $B'+C'$

A partir de la siguiente tabla de verdad Responder las preguntas 9 y 10.

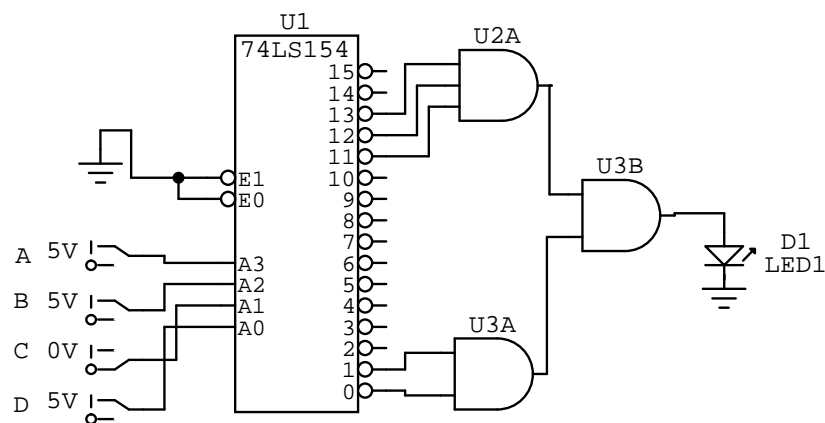
A	B	C	D	Y1	Y2	Y3	Y4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1
0	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1
0	1	0	1	1	1	1	1
0	1	1	0	1	1	1	1
0	1	1	1	1	0	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	1

9. Dado el siguiente circuito Función simplificada empleando multiplexor que corresponde a la Tabla de verdad es:



- a. Y1
- b. Y2
- c. Y3
- d. Y4

10. Dado el siguiente circuito Función simplificada empleando decodificador que corresponde a la Tabla de verdad es:



- e. Y1
- f. Y2
- g. Y3
- h. Y4

11. Los multivibradores monoestables son aquellos que:

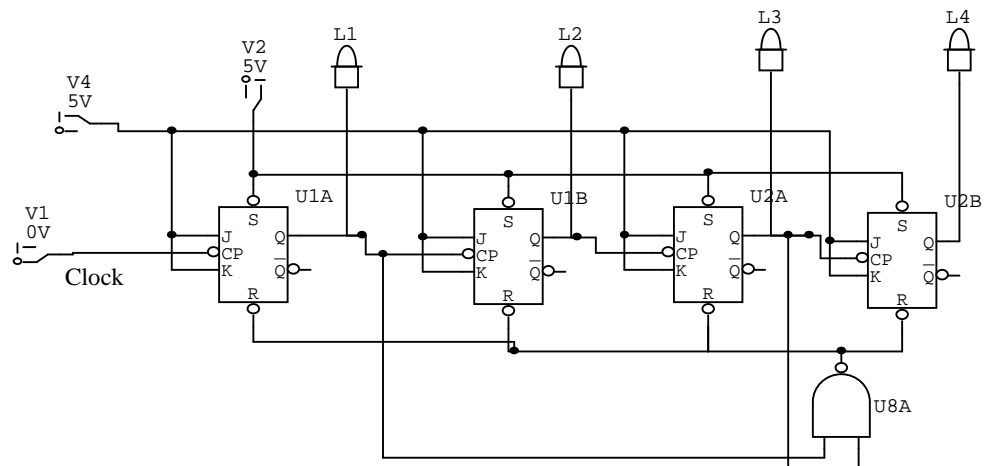
- a. Generan un pulso pequeño de corta duración cada vez que ocurre un cambio en su entrada
- b. Generan un tren de pulsos cada vez que ocurre un cambio en su entrada
- c. Generan un tren de pulsos sin necesidad de que ocurra un cambio en su entrada, ya que se conecta una red RC a esta.
- d. Genera un pulso pequeño de corta duración sin necesidad de que ocurra un cambio en su entrada, ya que se conecta una red RC a esta.

12. Los comparadores de magnitud de un bit se pueden realizar utilizando:

- a. Compuertas EXOR
- b. Compuertas NEXOR

- c. Todas las anteriores.
  - d. Ninguna de las anteriores.
13. Los Flipflops como elementos de memoria que almacenan un bit se construyen a partir de:
- a. Compuertas AND y OR
  - b. Compuertas NAND y NOR
  - c. Todas las anteriores.
  - d. Ninguna de las anteriores.
14. Los circuitos lógicos secuenciales se dividen básicamente en dos grupos: Los circuitos asincrónicos y los circuitos síncronos. Los primeros son:
- a. Aquellos que pueden cambiar los estados de sus salidas como resultado del cambio de los estados de las entradas.
  - b. Aquellos que pueden cambiar el estado de sus salidas en instantes de tiempo discretos bajo el control de una señal de reloj.
  - c. Todas las anteriores.
  - d. Ninguna de las anteriores.
15. Los circuitos lógicos secuenciales síncronos son:
- a. Aquellos que pueden cambiar los estados de sus salidas como resultado del cambio de los estados de las entradas.
  - b. Aquellos que pueden cambiar el estado de sus salidas en instantes de tiempo discretos bajo el control de una señal de reloj.
  - c. Todas las anteriores.
  - d. Ninguna de las anteriores.

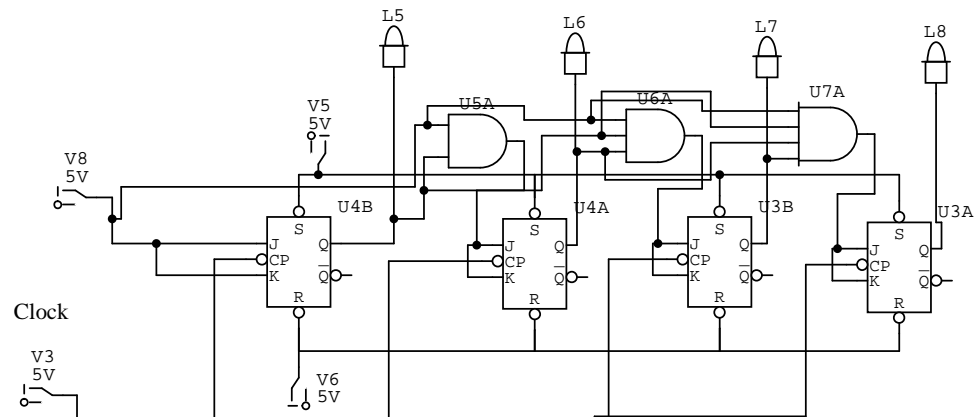
Dado el siguiente circuito responder las preguntas 16 y 17.



16. El circuito lógico secuencial mostrado en la figura es:
- a. Contador ascendente asíncrono reinicial de 0 a 4.
  - b. Contador ascendente asíncrono reinicial de 0 a 5.
  - c. Contador ascendente asíncrono reinicial de 0 a 6.
  - d. Contador ascendente asíncrono reinicial de 0 a 15.
17. Si deseamos obtener un contador descendente al conectar las salidas  $Q'$  ( $Q$  Negadas) obtendríamos:
- a. Contador descendente asíncrono reinicial de 15 a 11.
  - b. Contador descendente asíncrono reinicial de 15 a 10.
  - c. Contador descendente asíncrono reinicial de 15 a 9.

d. Contador descendente asíncrono reinicializable de 15 a 0.

Dado el siguiente circuito responder las preguntas 18 y 19.



18. El circuito lógico secuencial síncronico mostrado en la figura para que funcione correctamente debo:

- Conectar las entradas J y K a 5V.
- Conectar las entradas J y K a GND.
- Conectar la entrada J a 5V y K a GND.
- Conectar la entrada J a GND y K a 5V.

19. Para configurar el contador de forma descendente debo:

- Cambiar la conexión de las compuertas AND a las entradas Q'.
- Cambiar las compuertas AND por compuertas OR.
- Cambiar las compuertas AND por compuertas OR y conectarlas a las entradas Q'.
- Ninguna de las anteriores.

20. Las memorias EEPROM son aquellas que:

- Se programan Una única vez y sus datos no se pueden eliminar.
- Se programan por el Fabricante y sus datos se pueden borrar bien sea eléctricamente o por luz ultravioleta, pero no se les puede regrabar datos.
- Se programan por el Fabricante y sus datos se pueden borrar bien sea eléctricamente o por luz ultravioleta, pero se les pueden regrabar datos.
- No se programan por el Fabricante y sus datos se pueden borrar bien sea eléctricamente o por luz ultravioleta, pero se les puede grabar y regrabar datos.

**ANEXO 2 Respuestas Evaluación Diseño Electrónico Digital**

1	<b>a</b>	b	c	d
2	<b>a</b>	b	c	d
3	<b>a</b>	b	c	d
4	a	b	c	<b>d</b>
5	a	<b>b</b>	c	d
6	a	b	<b>c</b>	d
7	a	<b>b</b>	c	d
8	<b>a</b>	b	c	d
9	<b>a</b>	b	c	d
10	a	b	c	<b>d</b>
11	a	<b>b</b>	c	d
12	a	b	<b>c</b>	d
13	a	<b>b</b>	c	d
14	<b>a</b>	b	c	d
15	a	<b>b</b>	c	d
16	a	<b>b</b>	c	d
17	a	<b>b</b>	c	d
18	<b>a</b>	b	c	d
19	a	b	c	<b>d</b>
20	a	b	c	<b>d</b>



**ANEXO 3. PROYECTO CONVERSION ANALOGA DIGITAL****Objetivo General:**

Permitir que el estudiante pueda digitalizar señales físicas utilizando los componentes adecuados de Electrónica Digital.

El proyecto consiste en que el estudiante deberá adquirir una señal análoga (temperatura) por medio de un conversor Análogo Digital ADC0804, visualizar su digitalización a través de diodos Led e implementar un circuito que determinará el estado actual de la temperatura y dos (2) estados de alarma así:

- 1) Alarma A: Se activará cuando la temperatura este por debajo de los 12 grados Centígrados
- 2) Alarma B: Se activará cuando la temperatura supere los 38 grados Centígrados.

**Preguntas:**

- ¿Cuál es la resolución del conversor análogo digital?
- ¿Cuántas muestras por segundo puede tomar el ADC0804?
- ¿Qué modificaciones realizaría en el circuito para poder adquirir más de una señal?
- ¿Cuáles son las limitantes del sensor de temperatura?
- ¿Qué puede decir de la linealidad del sensor?

Elaborar informe que contenga: diagrama del circuito, respuestas a las preguntas anteriores, conclusiones y posibles aplicaciones para dicho montaje.

---